

# **BACHELORARBEIT**

## **Evolution von Kooperation in Roboterschwärmen**

von

**Christian Nagel**

**eingereicht am 14.09.2010 beim  
Institut für Angewandte Informatik  
und Formale Beschreibungsverfahren  
des Karlsruher Instituts für Technologie**

**Referent: Prof. Dr. Hartmut Schmeck**

**Betreuer: Lukas König**

Heimatanschrift:

Mainstraße 4

73529 Schwäbisch Gmünd

Studienanschrift:

Hirschstraße 23

76133 Karlsruhe



# Abstract

In der folgenden Arbeit wird die Evolution von Kooperation in Schwärmen von Robotern untersucht. Als Controller-Architektur kommen dabei künstliche neuronale Netze (KNN) zum Einsatz. Diese werden zunächst in das Java-Framework Easy Agent Simulation (EAS) implementiert, einer Weiterentwicklung des Finite Moore Generator (FMG) von Lukas König [KS08].

Die grundlegende Idee dieser Arbeit lässt sich wie folgt beschreiben: Auf dem Spielfeld befinden sich zwei verschiedene Arten von Individuen, zum einen Raubtiere, zum anderen Beutetiere. Ziel beider Arten ist es — wie auch in der Realität — ihr eigenes Überleben sowie den Fortbestand der Art zu sichern. Um nicht zu verhungern, müssen die Raubtiere Nahrung in Form von Beutetieren aufnehmen. Die Beutetiere hingegen versuchen dies zu verhindern. Sie flüchten vor den Raubtieren, um nicht gefressen zu werden.

Die Raubtiere werden mit einer fest implementierten Strategie versehen. Diese sieht vor, dass in regelmäßigen Abständen ein Beutetier ausgewählt wird. Dieses wird solange verfolgt, bis die Auswahl eines anderen Individuums ansteht (gesteuert durch einen Parameter). Die Auswahl eines neuen Ziels erfolgt zufällig aus all jenen Beutetieren, die sich in einem bestimmten Beute-Erkennungsradius um den Räuber herum befinden.

Jedes einzelne Beutetier wird durch ein eigenes künstliches neuronales Netz gesteuert. Zu Beginn besitzt dies keine beabsichtigte Funktionalität. Die neuronalen Netze werden am Anfang mutiert, um nicht mit vollständig leeren neuronalen Netzen zu starten. Die Funk-

tion des Netzes soll sich im Laufe der Evolution entwickeln. Um den Evolutionsprozess zu steuern, wird nicht — wie meist üblich — eine Fitness-Funktion verwendet. Stattdessen soll das Konzept der impliziten Fitness zum Tragen kommen.

Die Beutetiere können ihre eigenen Überlebenschancen erhöhen, indem sie miteinander kooperieren. Dazu sollen sie sich zu Schwärmen zusammenfinden, was dazu führt, dass die Raubtiere, die in bestimmten Zeitabständen ein zufälliges Beutetier in ihren Erkennungsradien auswählen, durch ständiges Wechseln insgesamt weniger Beutetiere fangen.

# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>7</b>
<b>2</b>	<b>Grundlagen</b>	<b>19</b>
2.1	Evolutionäre Algorithmen . . . . .	19
2.1.1	Theorie . . . . .	19
2.1.2	Ein Anwendungsbereich: Die Evolutionäre Robotik . . . . .	22
2.2	Neuronale Netze . . . . .	25
2.2.1	Grundprinzipien . . . . .	25
2.2.2	Verwendung in der Evolutionären Robotik . . . . .	30
<b>3</b>	<b>Implementierung</b>	<b>33</b>
3.1	Einordnung des implementierten Modells . . . . .	33
3.2	Simulationsumgebung . . . . .	36
3.3	Szenario . . . . .	38
3.4	Roboter . . . . .	40
3.4.1	Räuber . . . . .	40
3.4.2	Beute . . . . .	44
3.5	Weitere Implementierungen . . . . .	48
3.5.1	Statistik-Plugin . . . . .	48
3.5.2	Export von neuronalen Netzen . . . . .	49

<b>4</b>	<b>Experimente</b>	<b>51</b>
4.1	Methodik der Evaluation . . . . .	51
4.2	Auswertung/Ergebnisse . . . . .	55
<b>5</b>	<b>Zusammenfassung und Ausblick</b>	<b>61</b>
5.1	Zusammenfassung . . . . .	61
5.2	Ausblick . . . . .	62
<b>A</b>	<b>Parametersätze</b>	<b>65</b>

# Abbildungsverzeichnis

1.1	Das Prinzip des Darwinschen Evolutionsparadigmas . . . . .	10
1.2	Veranschaulichung des Schäfer-Problems . . . . .	15
2.1	Typische Vorgehensweise eines Evolutionären Algorithmus . . . . .	22
2.2	Aktivierungsfunktion des neuronalen Netzes . . . . .	27
2.3	Beispielhaftes evolviertes Netz in der Hinton-Darstellung . . . . .	28
2.4	Beispielhaftes neuronales Netz . . . . .	29
3.1	Visualisierung der Umgebung in der Simulation . . . . .	38
3.2	Visualisierung der Agenten in der Simulation . . . . .	41
3.3	Distanz- und Erkennungssensoren der Wolf-Roboter . . . . .	44
3.4	Visualisierung des anfänglichen neuronalen Netzes . . . . .	46
3.5	Distanzsensoren der Beute-Roboter . . . . .	47
4.1	Schaubilder aus Versuch 279, Versuchsreihe 14 . . . . .	56
4.2	Trajektorienbilder aus Versuch 279, Versuchsreihe 14 . . . . .	56
4.3	Schaubilder aus Versuch 202, Versuchsreihe 11 . . . . .	57
4.4	Trajektorienbilder aus Versuch 202, Versuchsreihe 11 . . . . .	58
4.5	Relative Häufigkeitsverteilung der Anzahl von Synapsen und von inneren Neuronen der am Ende des Simulationslaufs 279 evolvierten Netze . . . . .	60



# Tabellenverzeichnis

2.1	Evolutionäre Standardoperatoren: Rekombinationsoperatoren . . . . .	23
2.2	Evolutionäre Standardoperatoren: Mutationsoperator . . . . .	24
2.3	Evolutionäre Standardoperatoren: Selektionsoperatoren . . . . .	24
2.4	Lernverfahren in neuronalen Netzen . . . . .	31
4.1	Übersicht über die durchgeführten Experimente und deren Parameter . . .	54
A.1	Allgemeiner Standardparametersatz . . . . .	65
A.2	Spezieller Parametersatz des Master-Schedulers . . . . .	66
A.3	Spezieller Parametersatz des neuronalen Netzes . . . . .	66
A.4	Spezieller Parametersatz des Statistik-Moduls . . . . .	67



# Kapitel 1

## Einleitung

Die Anfänge der Evolutionstheorie lassen sich zu Beginn des 19. Jahrhunderts ausmachen. Obwohl einzelne Ideen zur Evolution schon lange Zeit vorher existierten, gilt Jean Baptiste de Lamarck als einer der ersten Verfasser eines in sich schlüssigen Evolutionskonzeptes. In seinem Werk „Philosophie Zoologique“ [Lam09] stellte er 1809 die These auf, dass die unterschiedlichen Arten durch die Vererbung von Anpassungen entstanden seien. Damit wandte er sich entschieden gegen die in seiner Zeit vorherrschende Annahme der Konstanz aller Arten. Er vertrat die Meinung, der Gebrauch eines Organs führe zu dessen funktioneller Verbesserung, wohingegen der Nichtgebrauch die Verkümmern zur Folge habe. Diese im Laufe eines Lebens mehr oder minder stark ausgeprägten Eigenschaften würden durch die Eltern an den Nachwuchs weitervererbt.

Eines der Beispiele aus der Natur, das Lamarck anführt, ist das der Giraffe. Ursprünglich habe sie über einen vergleichbar kurzen Hals wie andere Lebewesen verfügt. Die spärlich vorhandene Nahrung in Form von Blättern habe es jedoch notwendig werden lassen, den Hals zu strecken, um auch an Blätter heranzukommen, die an höheren Bäumen und Sträuchern wachsen. Über viele Generationen habe sich deswegen ein immer länger werdender Hals entwickelt. Als Triebkräfte einer derartigen Evolution proklamierte er zwei

Faktoren: Zum einen sei dieser Entwicklungsgang Ergebnis ungerichteter Anpassungen an äußere Veränderungen. Zum anderen spiele ein gewisser innerer Vervollkommenungstrieb eines jeden Individuums eine wichtige Rolle. Dieser führe zu immer komplexeren Lebewesen. Ein oft geäußelter Kritikpunkt an seiner Theorie ist das nicht hinreichend erklärbare Aussterben mancher Arten. Darauf reagierte er jedoch mit dem Hinweis, dass die Art unter Umständen in einem anderen Teil der Welt noch lebe oder dass sich die Art so sehr gewandelt habe, dass man sie nicht mehr erkennen könne. Lamarck erntete durch seine avantgardistischen Ideen zunächst jedoch nur Ablehnung.

Kontrovers diskutiert wurde bzw. wird auch Charles Darwin, der ein halbes Jahrhundert später im Jahr 1859 sein Werk „On the Origin of Species“ [Dar59] veröffentlichte<sup>1</sup>. Eine von Darwins Hypothesen lautet, dass sich Individuen, die besser an die Umwelt angepasst sind, häufiger vermehren im Vergleich zu denjenigen, die weniger gut an die äußeren Gegebenheiten angepasst sind (Prinzipien der natürlichen Auslese durch Selektion sowie der Vererbung). Ein weiterer Bestandteil von Darwins Theorie ist das Axiom des Gradualismus. Damit beschreibt er, dass die Übergänge zwischen den einzelnen Umweltzuständen allmählich und in kleinen Schritten verlaufen. Dies ermögliche auch erst eine Anpassung an veränderte Umweltbedingungen, die bei abrupten Umweltveränderungen nicht möglich sei [KEH00, Seite 351].

Eine Erklärung für die genaue Funktionsweise der Vererbung blieb Darwin jedoch schuldig. Diese lieferte erst Gregor Mendel wenige Jahre später, nämlich 1865, mit seinen in großem Umfang durchgeführten systematischen Kreuzungsexperimenten an Erbsenpflanzen, durch die er die nach ihm benannten Mendelschen Regeln entdeckte. Seine 1866 veröffentlichten „Versuche über Pflanzenhybriden“ [Men66] trafen jedoch auf Unverständnis in der damaligen Fachwelt und fanden zunächst keine größere Beachtung [KEH00, Seite 256].

---

<sup>1</sup>Alfred Russel Wallace entwickelte unabhängig davon bereits 1857 eine ähnliche Theorie

Das Gedankengebäude von Darwin wurde seit dem 19. Jahrhundert ständig erweitert. Etwas mehr als ein halbes Jahrhundert nach Mendels Entdeckungen wurden dessen Erkenntnisse 1930 mit Darwins Theorie kombiniert, wodurch die Synthetische Theorie der Evolution entstand. Für deren Verständnis sind die Begriffe „Genotyp“ und „Phänotyp“ bedeutsam. Der Genotyp beschreibt die Eigenschaften eines Organismus auf der Ebene der Erbfaktoren, der Phänotyp hingegen befasst sich mit den äußerlich erkennbaren Eigenschaften einer Lebensform. Letzterer wird maßgeblich vom Genotyp beeinflusst — eine Erkenntnis, welche bisher in Darwins Theorie nicht vorhanden war. Nicht jede Änderung auf der Ebene des Genotyps muss sich allerdings auch auf die Phänotyp-Ebene auswirken. Zufallsveränderungen, also Mutationen, können den Genotyp und damit den Phänotyp verändern. Dieser durch Mutation veränderte Organismus wird dem natürlichen Selektionsprinzip ausgesetzt und muss sich nun durchsetzen, um in den folgenden Generationen weiter Bestand haben zu können.

Das Prinzip des Darwinschen Evolutionsparadigmas (vgl. Abbildung 1.1) machen sich auch zunehmend andere Bereiche außerhalb der Biologie zu Nutze. So gab es Ende der fünfziger Jahre des 20. Jahrhunderts bereits erste Ansätze zu Evolutionären Algorithmen. In den folgenden Jahrzehnten bildeten sich unterschiedliche Schulen heraus. Es entstanden die Richtungen der Genetischen Algorithmen, deren wesentliche Eigenschaften eine wahrscheinlichkeitsbasierte Elternselektion ist und die Rekombination als primären Suchoperator nutzt. Der Genotyp der Individuen ist beim Ansatz der Evolutionsstrategien grundsätzlich reellwertig, es herrscht kein Selektionsdruck bei der Elternauswahl und nur die besten Individuen überleben in der Umweltselektion. Primärer Operator ist hier die Mutation. Ferner entstand die Richtung des Evolutionären Programmierens, deren Grundidee darin besteht, die Evolution auf eine eher verhaltensbasierte Ebene zu verlagern. Dabei ist der Phänotyp von vorrangigem Interesse, ein Rekombinationsoperator existiert in der Regel nicht. Weiter ist die Herangehensweise des Genetischen Programmierens zu

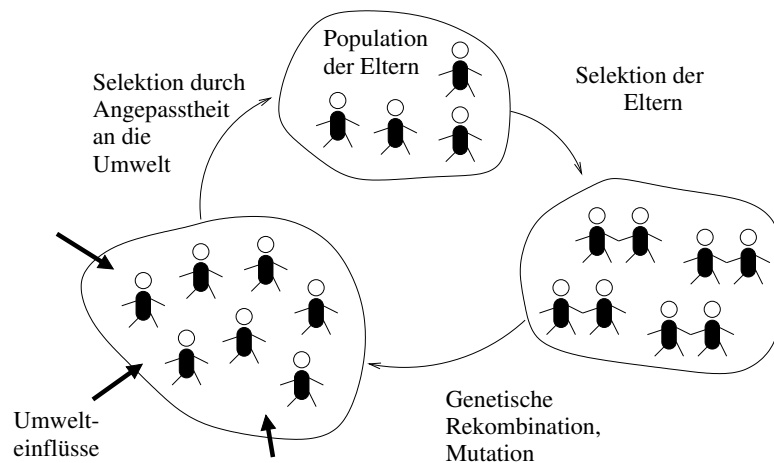


Abbildung 1.1: Das Prinzip des Darwinschen Evolutionsparadigmas nach [Wei99]

erwähnen, die im Kontext Evolutionärer Algorithmen entstanden ist. Hier ist der Hauptoperator die Rekombination, weiter wird eine variable Größe der jeweiligen Repräsentation unterstellt. Der ursprüngliche Kerngedanke war, Programme bzw. Funktionen zu evolvieren, die als Syntaxbäume dargestellt werden können. Für eine detaillierte Diskussion dieser Begriffe sei auf [Wei99] verwiesen.

Ende des 20. Jahrhunderts einigte man sich auf den Oberbegriff der Evolutionären Algorithmen. Im Laufe der Zeit haben sich eine große Anzahl an weiteren Standardalgorithmen herausgebildet. Der Sammelbegriff „Evolutionäre Algorithmen“ umfasst eine Vielzahl von Verfahren, die bewusst Prinzipien der Evolution nachahmen. Ziel dabei ist es, eine Such- oder Optimierungsaufgabe approximativ zu lösen oder — ausgehend von einer gegebenen Lösung — zu einer besseren zu gelangen. Ein Anwendungsgebiet hierfür findet sich in der Evolutionären Robotik. Aufgabe dieses Forschungsgebietes ist es, Steuerprogramme für autonome mobile Roboter zu entwickeln, welche durch Anwendung des evolutionären Prinzips ein bestimmtes erwünschtes Verhalten lernen sollen.

Die Schwierigkeit der zu lösenden Aufgaben wächst stetig. Der Entwurf von Steuerprogrammen, welche dieser Entwicklung Rechnung tragen, gestaltet sich schwierig („design

problem“). Dies führt dazu, dass die Controller nicht mehr von Hand programmiert werden können, da dies entweder zu aufwändig ist oder man schlichtweg über kein hinreichend großes Vertrauen in die so entwickelten Lösungen verfügt.

Ein weiteres Problem liegt in der Tatsache begründet, dass das erwünschte Verhalten auf Ebene des gesamten Roboterschwarms charakterisiert wird; programmiert werden sollen allerdings die einzelnen Roboter des Schwarms. Die Vorhersage des Globalverhaltens von Schwärmen ist jedoch nicht einfach, da sich dieses Verhalten nicht allein dadurch vorhersagen lässt, dass man die Einzelleistungen addiert und Synergieeffekte vernachlässigt. Kollektives Verhalten in Roboterschwärmen setzt sich sowohl aus der Interaktion der Individuen untereinander als auch der Interaktion mit der (meist auch noch dynamisch veränderbaren) Umwelt zusammen. Um diese Unwägbarkeiten zu lösen, möchte man den Umgang mit der beschriebenen Komplexität dem verwendeten Lernverfahren (also hier der Evolution) überlassen und auf diese Weise erreichen, dass die Individuen sich an die gestellte Aufgabe anpassen, sich deren Lösung annähern und im Optimalfall ein globales Optimum erreichen. Dies stellt auch die Motivation der folgenden Arbeit dar.

Einsatzgebiete, in denen die Ergebnisse der Evolutionären Robotik potentiell von großem Interesse sein können, sind insbesondere dynamische Umgebungen, die dem Menschen nicht vollständig bekannt oder für ihn nicht zugänglich sind. Beispiele sind die Erforschung unbekannter Regionen wie z.B. die des Weltalls oder die anderer Planeten, Einsätze in der Medizintechnik, mit welcher man hofft, in ferner Zukunft Mini-Roboter durch Injektion in den menschlichen Körper zur Heilung von Krankheiten einsetzen zu können oder der Einsatz zur Erkundung von und die Aufgabenerfüllung in Katastrophengebieten [SKM10]. An aktuellen Beispielen mangelt es kaum, so könnte man im Jahr 2010 die Versiegelung des Öllecks infolge der Ölkatastrophe im Golf von Mexiko anführen, eine Aufgabe, die — sollte sie in dieser oder einer ähnlichen Form wieder auftreten — in fernerer Zukunft von autonomen mobilen Robotern erledigt werden könnte.

Unter dem Begriff der Kooperation verstehen wir dem allgemeinen Sprachgebrauch nach ein Zusammenwirken von zwei oder mehreren Individuen oder Systemen auf ein gemeinsames Ziel hin. Prägnant formuliert handelt es sich um (gegenseitige) Hilfe. Weiter wird in Kooperation mit und ohne Kosten unterteilt. Sobald einem Individuum Kosten entstehen, spricht man von starker Kooperation oder Altruismus.

Der Begriff der Kooperation stellt auf den ersten Blick für das Darwinsche Evolutionsparadigma eine Schwierigkeit dar [AH81]. Dieses sieht Kooperation nur dann vor, wenn sie zu einem effektiveren Kampf um das eigene Überleben führt. Aus welchem Grund sollte man mit einem anderen Individuum kooperieren und ihm behilflich sein, wenn dies augenscheinlich nicht zu einer höheren Überlebenswahrscheinlichkeit führt? In der Tierwelt gibt es einige Phänomene, die sich nicht mit der ursprünglichen Darwinschen Evolutionstheorie erklären lassen (beispielsweise altruistische Verhaltensweisen). Daher wurden zwei Arten von Erweiterungen für diese evolutionäre Theorie entwickelt: Zum einen ist dies die Theorie der Verwandtenselektion („theory of kin selection“ [Ham64]) und zum anderen die Theorie des reziproken Altruismus („theory of reciprocal altruism“ [Tri71]).

Die Theorie der Verwandtenselektion besagt, dass Altruismus die Reproduktion positiv beeinflussen kann, wenn die Individuen nah genug miteinander verwandt sind. Dies liegt darin begründet, dass die Wahrscheinlichkeit höher ist, dass nahe Verwandte die gleichen (oder zumindest ähnliche) Gene in sich tragen. Durch stark kooperative oder altruistische Verhaltensweisen lässt sich so der Anteil von eigenen Erbfaktoren im Genpool (also der Gesamtheit aller Genvariationen einer Population) erhöhen. Dies deckt sich mit der Beobachtung, dass sich die meisten selbstlosen Handlungen im Tierreich zwischen nahen Verwandten abspielen. Als anschauliches Beispiel aus der Natur ist der Stich einer Arbeiterbiene zu nennen, die unter Einsatz ihres eigenen Lebens versucht, ihr Bienenvolk zu verteidigen. In diese Kategorie lassen sich die meisten koloniebildenden Lebewesen einordnen, die selbstlose Verhaltensweisen an den Tag legen.

Die Theorie des reziproken Altruismus versucht die Beobachtung zu beschreiben, dass sich kooperative Handlungen auch zwischen weit entfernt oder überhaupt nicht miteinander verwandten Individuen ausbilden können. Sie basiert auf der Annahme, dass selbstlose Verhaltensweisen eines Individuums A gegenüber Individuum B zu einem späteren Zeitpunkt Individuum B dazu veranlassen, altruistisch gegenüber Individuum A zu handeln. Ein Beispiel aus der Natur ist die Flechte, ein Mutualismus (also das gemeinsame Auftreten) zwischen Alge und Pilz. Des Weiteren das Verhältnis zwischen bestimmten Ameisenarten und Akazienbäumen, bei dem die Ameisen in hohlen Dornen leben und mit Nektar versorgt werden. Im Gegenzug sorgen die Insekten dafür, dass die Akazie nicht von Schädlingen befallen wird. Die gegenseitige Fellpflege bei Primaten ist ein weiteres Beispiel für eine reziproke altruistische Verhaltensweise im Tierreich [CR06, Seite 1367-1370].

Aus betriebswirtschaftlicher Sicht ist der Begriff der Kooperation laut Wöhe durch eine „freiwillige Zusammenarbeit von Unternehmen, die [...] rechtlich selbstständig bleiben, gekennzeichnet.“ Weiter erfolgt „die Zusammenarbeit [...] i.d.R. zu dem Zweck, [...] die Wettbewerbsfähigkeit zu steigern.“ [WD08, Seite 255]. Zentraler Aspekt dieser Definition ist also die Zusammenarbeit mehrerer Unternehmen auf ein gemeinsames Ziel hin, von dem alle Beteiligten profitieren. Über diese Definition einer Kooperation hinaus finden wir auch weitere kooperative Formen wie z.B. das Prinzip der Arbeitsteilung, ohne welche die heutige Wirtschaftswelt nicht denkbar wäre.

Die Soziobiologie beschreibt Kooperation als ein Teil des Sozialverhaltens von Lebewesen, bei welchem sich die Beteiligten gegenseitig nützlich sind. Es wird dadurch hervorgerufen, dass ein bestimmtes Verhalten in der Gruppe zu größerem Erfolg führt als bei einem Einzeltier. Kooperation ist verbunden mit der Bildung eines Systems auf höherer Ebene, welches auf eine bestimmte Art und Weise nach Verbesserung strebt [CR06, Seite 1358].

Sowohl zwischen gleichen als auch zwischen unterschiedlichen Arten scheinen sich Mechanismen der Kooperation als Prinzip im Laufe der Evolution bewährt zu haben. Auch in der

heutigen hart umkämpften Wirtschaftswelt lässt sich dieser Grundsatz vorfinden. Daher soll in dieser Arbeit der Fragestellung nachgegangen werden, ob ein solch grundlegendes Prinzip auch im Rahmen einer bestimmten Aufgabe in Roboterschwärmen evolvierbar ist.

Kooperation wird im Folgenden verstanden als Bildung eines Schwarms von Beutrobotern, die sich dadurch besser vor Räuberrobotern schützen lernen sollen. Im Speziellen soll die Eigenschaft des Räubers, in bestimmten Zeitabständen ein neues Beutetier auszuwählen, ausgenutzt werden, sodass insgesamt weniger Beutetiere gefangen werden. Diese Fähigkeit zur Kooperation soll im Laufe von mit unterschiedlichen Parametern durchgeführten Evolutionsläufen im Szenario einer klassischen Räuber-Beute-Beziehung erprobt werden.

Die einzelnen Evolutionsläufe werden in Simulationen durchgeführt. Damit lässt sich einerseits Zeit sparen, da Experimente mit realen Robotern wesentlich zeitaufwändiger sind. Auch lassen sich die Evolutionsläufe auf mehreren Rechnern parallelisieren und somit insgesamt mehr Roboter testen. Weiterhin entfallen typische Probleme wie z.B. die kostenintensive Bereitstellung einer großen Anzahl von realen Robotern, deren Energieversorgung, das Zurücksetzen der Umgebung nach jedem Evolutionslauf und ähnliche zu berücksichtigende Faktoren. Weiterhin lassen sich verschiedene Anforderungen in unterschiedlichen Experimenten schnell umsetzen. Nachteile dabei sind jedoch, dass einerseits Probleme gelöst werden müssen, die sich in Realität nicht stellen (Programmierung der Umgebung, der Sensoren etc.). Des Weiteren lassen sich die Ergebnisse aus Simulationen nicht eins zu eins in die Realität übertragen, weil die Roboter minimale Unterschiede (z.B. der Sensorik) aufweisen können, welche zur Funktionsunfähigkeit des in Simulationen evolvierten Steuerprogramms führen können [NF01, Seite 57-58]. Grundlegende Prinzipien der Evolution lassen sich jedoch durchaus übertragen [WGW03]. Auch wenn einige negative Aspekte vorhanden sind, so überwiegen doch die Vorteile, weshalb diese Arbeit sich ausschließlich auf Simulationsergebnisse stützt.

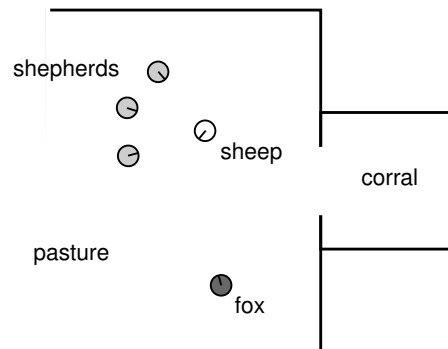


Abbildung 1.2: Veranschaulichung des Schäfer-Problems aus [PMS01]

In [WKF09] wurden Experimente durchgeführt, aus denen hervorgeht, dass es mit steigenden Kooperationsanforderungen in der Regel sinnvoller ist, auf homogene Gruppen von Robotern zurückzugreifen. Allerdings ist auch darauf hinzuweisen, dass dadurch unter Umständen vorzeitig eine suboptimale Lösung erreicht wird, die auf lange Sicht von heterogenen Teams, bei denen einen größerer Suchraum durchsucht wird, übertroffen werden könnte. Da es sich hierbei jedoch um eine nicht durch Studien abgesicherte Vermutung handelt, werden im Folgenden ausschließlich homogene Gruppen (also Gruppen von Individuen, deren Genotyp sich zu Beginn der Simulation nicht unterscheidet) verwendet.

Nun folgen exemplarisch einige verwandte Untersuchungen. In [PMS01] wird das sogenannte Schäfer-Problem („herding problem“, vgl. Abbildung 1.2) untersucht. Bei dieser Aufgabenstellung soll eine Gruppe von Robotern (Schäfer) einen anderen Roboter (Schaf) in einen vordefinierten Bereich treiben. Das Schaf versucht, dies zu verhindern und den Schäfern zu entkommen. Die Aufgabe wird zusätzlich erschwert durch einen weiteren Roboter (Fuchs), der nach dem Leben des Schafs trachtet und vor dem die Schäfer das Schaf beschützen müssen. Während Schaf und Fuchs fixe Strategien haben, werden die Schäfer durch ein neuronales Netz gesteuert. Das Ergebnis ist, dass eine Kooperation der Schäfer durch Spezialisierung erreicht wird. Spezialisierung auf bestimmte Aufgaben kann jedoch

nur entstehen, wenn die Aufgabe relativ schwierig ist und mehrere Fähigkeiten zur Lösung der Aufgabe notwendig sind [PMS01].

Ein weiteres Problem, das in diesem Zusammenhang häufig untersucht wird, ist das sogenannte Transport-Problem. Dabei ist das Ziel der Roboter, eine Box in ein bestimmtes Gebiet zu schieben. Dazu sollen sie sich zu kleinen Gruppen zusammenschließen und kooperieren, da die Box für einen Roboter alleine zu groß ist und er folglich nicht in der Lage ist, die Box zu bewegen. Die Aufgabe kann ohne eine Zusammenarbeit nicht gelöst werden. Bei der Untersuchung dieses Problems identifizierten Zhang und Cho [ZC99] folgende zusammengesetzte Kooperationsstrategie: Zunächst bilden die Roboter eine Gruppe in der Nähe der Box, begeben sich anschließend gemeinsam zur Box und schieben diese danach weiter im Schwarm auf den Zielbereich zu, bevor sie kooperativ ihr Ziel erreichen.

Die den Experimenten zugrunde liegende Implementierung hat Ähnlichkeiten zu „Cascade Correlation“ und zu „NeuroEvolution of Augmenting Topologies“. Auf diese beiden Ansätze wird im Implementierungskapitel (Kapitel 3) näher eingegangen.

Der hier verwendete evolutionäre Ansatz lässt sich durch die zwei Schlagwörter „online“ und „onboard“ (auch: „dezentral“) charakterisieren. Dabei meint „online“, dass eine Aufgabe von einem Agenten in Echtzeit zu erfüllen ist; die potentielle Lösung für das gegebene Problem wird — ebenfalls in Echtzeit — hinsichtlich ihrer Zweckmäßigkeit evaluiert und gegebenenfalls korrigiert. Der Begriff „onboard“ besagt in diesem Zusammenhang, dass jeder einzelne Roboter autark agiert und nicht durch eine zentrale Instanz gesteuert wird. Das heißt, das Individuum kann nur auf die ihm zur Verfügung gestellten Sensoren, seine eigene Rechenkraft und die eigenen Kommunikationsmöglichkeiten zurückgreifen, die ihm zugestanden werden [KS08].

Das Räuber-Beute-Szenario wird hauptsächlich in Arbeiten zur Koevolution herangezogen. In den hier durchgeführten Experimenten handelt es sich ausdrücklich nicht um Experimente der Koevolution (vgl. auch [Jan80]). Nach Campbell und Reece [CR06, Seite

1411] bezieht sich der Begriff „Koevolution“ auf eine „reziproke evolutionäre Anpassung zwischen zwei Arten. Durch die Veränderung einer Art wird ein Selektionsdruck auf eine andere ausgeübt, und diese Gegenanpassung fördert wiederum die evolutionäre Abwandlung der ersten Art“. Der Räuber hat in der vorliegenden Arbeit eine feste Strategie und ist daher nicht fähig, auf Strategieanpassungen seitens der Beute zu reagieren. Ein gegenseitiges „Wettrüsten“ („arms race“), wie Dawkins und Krebs [DK79] es festgestellt haben, kann somit nicht entstehen.

Wie bereits Hoen u.a. feststellten, kann die Klassifikation eines bestimmten Szenarios im Einzelfall schwierig und nicht eindeutig sein [HTP<sup>+</sup>06, Seite 3]. So können sowohl kompetitive als auch kooperative Aspekte in einer Aufgabenstellung zu finden sein. Eine kompetitive Aufgabenstellung zeichnet sich dadurch aus, dass die Ziele der einzelnen Individuen entgegengerichtet sind. Einzelne Individuen profitieren auf Kosten anderer. Bei kooperativen Problemen sind die Ziele hingegen gleichgerichtet. Alle Mitglieder einer Population sind dort entweder zusammen erfolgreich oder gehen gemeinsam unter. Die Veränderung eines Individuums — sowohl in positiver als auch in negativer Richtung — geht dort mit einer Verbesserung für die anderen Individuen einher [PL05, Seite 391]. Das klassische Räuber-Beute-Szenario weist damit zunächst vor allem kompetitive Züge auf. So sind die Ziele von Räuber (Fangen der Beute) und Beute (Vermeiden, gefangen zu werden) stets entgegengerichtet. Das im Folgenden untersuchte Szenario weist allerdings auch kooperative Aspekte auf. So können die Beutetiere durch Zusammenarbeit in Form einer Schwarmbildung die Wahrscheinlichkeit senken, gefangen zu werden. Obgleich dieses Szenario sowohl kompetitive als auch kooperative Züge aufweist, soll der Fokus dieser Arbeit vor allem auf der Evolution von kooperativen Möglichkeiten innerhalb der Beutepopulation liegen. Damit wird keine Koevolution im eigentlichen Sinne untersucht. Vielmehr handelt es sich um eine Untersuchung einer kooperativen Aufgabenstellung in einem koevolutionären Szenario.

Die Arbeit gliedert sich folgendermaßen: In Kapitel 2 werden die dem Verständnis der Arbeit unabdingbaren Grundlagen gelegt, bevor im dritten Kapitel zunächst auf die Implementierung eingegangen wird. Hier findet sich sowohl eine Beschreibung aller erstellten Implementierungen im bestehenden EAS-Framework als auch eine Illustrierung der zu lösenden Aufgabenstellung. Anschließend werden in Kapitel 4 die durchgeführten Experimente beschrieben, verwendete Auswertungsverfahren vorgestellt und auf die einzelnen Simulationsläufe angewendet. Kapitel 5 beschäftigt sich mit einer abschließenden Zusammenfassung der Ergebnisse und gibt einen Ausblick auf zukünftig mögliche weitergehende Fragestellungen. Im Anhang werden die den Simulationsläufen zu Grunde liegenden Parametersätze präsentiert und erläutert.

# Kapitel 2

## Grundlagen

Dieses Kapitel befasst sich mit den theoretischen Grundlagen dieser Arbeit. Es ist wie folgt gegliedert. Zunächst wird in die Theorie der Evolutionären Algorithmen eingeführt, bevor ein Anwendungsbereich Evolutionärer Algorithmen, die Evolutionäre Robotik, näher beleuchtet wird. Anschließend werden die Grundprinzipien von neuronalen Netzen erläutert und deren Verwendung in der Evolutionären Robotik thematisiert. Die hier gemachten Ausführungen stellen die Basis für das Verständnis der folgenden Kapitel dar.

### 2.1 Evolutionäre Algorithmen

#### 2.1.1 Theorie

Bei evolutionären Algorithmen handelt es sich um eine Klasse von stochastischen Lösungsverfahren für Optimierungsprobleme. Sie zeichnen sich durch einen iterativen Ansatz aus und werden inspiriert vom natürlichen Evolutionsprinzip. Die Natur hat damit im Laufe von Jahrmillionen komplexeste Lebewesen geschaffen. Der beschränkte Zeitrahmen im Zusammenhang mit einer Lösung eines Optimierungsproblems erlaubt jedoch nur die Nachahmung dieses Prinzips in wesentlich kleinerem Umfang und in einer stark abstrah-

hierten Fassung. Die Evaluation der Güte eines Organismus in der Natur erfolgt stets implizit (d.h. durch eine natürliche Auslese), wohingegen diese Aufgabe bei evolutionären Algorithmen meist eine sogenannte Güte- oder Fitnessfunktion erfüllt. Diese Arbeit wagt — dem natürlichen Prinzip entsprechend — den Versuch, ohne explizite Fitnessfunktion auszukommen und dem Prinzip des „survival of the fittest“ folgend zu einem Ergebnis zu gelangen. Evolutionäre Operatoren versuchen, einzelne Effekte der Evolution nachzuahmen.

Eine Eigenschaft, welche Evolutionäre Algorithmen besonders attraktiv macht, liegt darin, dass über das eigentliche Problem nur wenige Informationen bekannt oder bestimmte Anforderungen erfüllt sein müssen; es genügt, wenn eine Darstellung des Suchraumes sowie eine Bewertungsfunktion für die einzelnen Lösungskandidaten existiert. Damit ist das Verfahren in verschiedensten Bereichen anwendbar. Weiter sind flexible Anpassungen an geänderte Anforderungen durch den iterativen Ansatz sogar noch während der Laufzeit des Algorithmus möglich.

Eine mögliche Lösung sei im Folgenden als Individuum verstanden. Die Menge aller Individuen in einem räumlich und zeitlich abgegrenzten Bereich wird als Population bezeichnet. Ein Individuum lässt sich auf Ebene des Geno- oder des Phänotyps beschreiben. Der Genotyp beschreibt die im Genom gespeicherte Information. Der Phänotyp beschreibt das von außen unterscheidbare Verhalten, hier also, wie gut ein Kandidat das gestellte Problem lösen kann.

Die Individuen mit höherer Fitness werden im evolutionären Prozess häufiger vervielfältigt als jene mit niedrigerer Bewertung. Dieser Schritt stellt die Verbindung zwischen den potentiellen Lösungskandidaten und dem Optimierungsproblem her. Um zu verhindern, dass statt einem globalen Maximum nur ein lokales Maximum erreicht wird, ist auf eine ausreichende Diversität innerhalb der Population zu achten. Der Begriff der Diversität lässt sich anschaulich mit einer möglichst nicht ungleichmäßigen Verteilung der Kandidaten im

Suchraum beschreiben. Dabei sollte die Diversität weder zu hohe noch zu niedrige Werte annehmen, denn bei der Wahl muss eine Abwägung zwischen „Exploration“ (Erkundung des gesamten Suchraumes) einerseits und „Exploitation“ (Durchsuchung der Umgebung von Individuen mit hoher Fitness) andererseits getroffen werden. Gewöhnlicherweise geht mit hoher Diversität ein eher niedriger Selektionsdruck einher, entsprechendes gilt umgekehrt. Um diese Kenngröße zu quantifizieren, lässt sich beispielsweise der Hamming-Abstand (Anzahl der Unterschiede im Genom) zwischen jeweils zwei Individuen berechnen und die einzelnen Werte über die gesamte Population summieren (oder alternativ das arithmetische Mittel bilden), d.h.

$$\sum_{i=1}^n \sum_{j=1, j \neq i}^n \text{hamming}(g_i, g_j) \quad (2.1)$$

wobei  $n$  die Populationsgröße und  $g_i$  das Genom des  $i$ -ten Individuums beschreibt.

In der vorliegenden Arbeit besteht die Menge der zu überprüfenden potentiellen Lösungskandidaten aus künstlichen neuronalen Netzen (KNN). Wie oben erwähnt, wird ein impliziter Bewertungsansatz verwendet. Künstliche neuronale Netze der Beuteroboter, die sich nicht bewähren, werden dadurch aus dem Pool aller Netze entfernt, indem jene bei Gefangenwerden nicht mehr an andere Individuen weitergegeben werden.

Die typische Vorgehensweise, um einen evolutionären Zyklus zu simulieren, ist laut Weicker [Wei07, Seite 24-25] folgende (vgl. auch Abbildung 2.1):

Zunächst wird eine Startpopulation erstellt (Initialisierung). Diese kann je nach Problemstellung zufällig oder problembezogen gewählt werden. Sie kann beispielsweise auch das Ergebnis eines anderen Optimierungsverfahrens sein, sofern dort noch keine optimale Globallösung gefunden wurde. Anschließend werden die neuen Individuen anhand der Gütefunktion in Bezug auf das Optimierungsproblem bewertet (Bewertung). Solange die Terminierungsbedingung nicht erfüllt ist, startet der iterative Prozess. Dieser führt wiederholt eine Auswahl der für die Kindbildung herangezogenen Individuen (Paarungsselektion), eine Durchmischung der Population (Rekombination) und eine Mutation durch. Bei

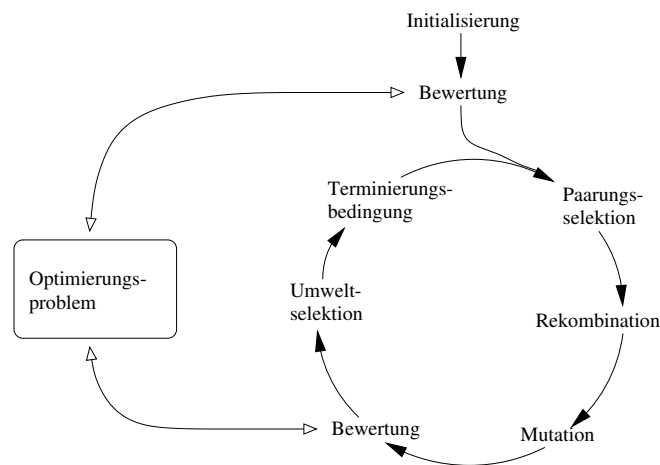


Abbildung 2.1: Typische Vorgehensweise eines Evolutionären Algorithmus nach [Wei99]

der Mutation werden in der Regel nur geringe Änderungen vorgenommen, um die Vererbung von Eigenschaften nicht zu stark zu behindern. Daraus geht die nächste Generation hervor, die wiederum der Bewertung hinsichtlich des Optimierungsproblems (Bewertung) sowie der Umweltselektion unterliegt. Das Optimierungsverfahren stoppt, wenn die Terminierungsbedingung erfüllt ist. Je nach Implementierung geschieht dies, sobald eine bestimmte Anforderung an den Lösungskandidaten erfüllt ist oder ein vorab bestimmter Zeitrahmen (angegeben beispielsweise als Anzahl an Iterationen) verstrichen ist.

In dieser Arbeit werden nur Mutationsoperatoren verwendet. Dies ist der Tatsache geschuldet, dass diese wesentlich einfacher zu implementieren sind. Für zukünftige Arbeiten sind jedoch durchaus auch Rekombinationsoperatoren denkbar. Häufig verwendete evolutionäre Standardoperatoren sind mit ihren Charakteristika zur Übersicht in den Tabellen 2.1, 2.2 und 2.3 zusammengefasst.

### 2.1.2 Ein Anwendungsbereich: Die Evolutionäre Robotik

Im Forschungsbereich der Evolutionären Robotik („Evolutionary Robotics“) besteht das Optimierungsproblem darin, dass eine vorab definierte Aufgabe durch ein bestimmtes

Tabelle 2.1: Evolutionäre Standardoperatoren: Rekombinationsoperatoren nach [Wei07, Seite 80-84].

Klassifikation	Erläuterung	Vorteile	Nachteile
kombinierend	Kombination von Eigenschaften der Eltern	vergleichsweise einfache Implementierung	keine neuen Teile des Suchraums erreichbar
interpolierend	Bildung von Zwischenwerten aus Eltern-eigenschaften	Abschwächung von Sprüngen im Suchraum (Dämpfung von Mutationsausreißern), Feinabstimmung	Population neigt zur Konvergenz gegen Schwerpunkt
explorativ	Evaluation vielversprechender Bereiche, basierend auf Prämissen über den Suchraum	Erhöhung der Vielfalt in der Population, Erforschung des Suchraums	Suchraum wird evtl. verlassen, Lösung muss wieder zulässig gemacht werden

Tabelle 2.2: Evolutionäre Standardoperatoren: Mutationsoperator nach [Wei07, Seite 114-115].

Vorgehen	Vorteile	Nachteile
zufälligkeitsbasierte Veränderung von Eigenschaften der Eltern	Erforschung des Suchraums, Feinabstimmung, Erhöhung der Diversität	u.U. negative Auswirkungen auf Konvergenz des Algorithmus bei falscher Parameterwahl

Tabelle 2.3: Evolutionäre Standardoperatoren: Selektionsoperatoren nach [Wei07, Seite 67].

Klassifikation	Erläuterung
stochastisch/ probabilistisch	Individuenauswahl fitness- und/oder zufallszahlenbasiert
deterministisch	Individuenauswahl rein fitnessbasiert

Verhalten eines oder mehrerer Roboter gelöst werden soll. Roboter werden als Einheiten gesehen, die ihr Verhalten im Wechselspiel mit der Umwelt evolvieren. Der Programmierer stattet die Roboter mit einer Menge an Grundfertigkeiten aus, aus denen sich im Laufe der Evolution einfache sowie komplexere Verhaltensweisen ausbilden sollen, welche der Aufgabenstellung genügen. Er schafft somit lediglich den Rahmen für die Evolution, indem er eine entsprechende Umgebung bereitstellt, (im Falle eines Realexperiments) die Roboter mit Energie versorgt usw. Sobald ein Evolutionslauf begonnen hat, ist der Mensch nur noch Kontrollorgan. Grundlegende, häufig evolvierte Aufgabenstellungen der Evolutionären Robotik sind Collision Avoidance oder Gate Passing. Bei Collision Avoidance geht es darum, Zusammenstöße mit Wänden, anderen Robotern oder ähnlichen Hindernissen zu vermeiden. Gate Passing versucht ein Verhalten entstehen zu lassen, bei dem ein auf dem Spielfeld platziertes Tor möglichst häufig durchfahren wird.

## 2.2 Neuronale Netze

### 2.2.1 Grundprinzipien

Das grundlegende Modell der Neuronen stammt aus den Arbeiten von McCulloch und Pitts [MP43]. Neuronale Netze lassen sich demzufolge als Ansammlung von Einheiten (Neuronen) beschreiben. Diese sind durch gewichtete Verbindungen (Synapsen) miteinander verbunden. Zweck der Neuronen ist es, empfangene Signale zu verarbeiten und weiterzuleiten. Die Eingabeneuronen empfangen dabei Signale aus der Umgebung, in der sich das neuronale Netz befindet. Ausgabeneuronen geben analog dazu Signale ab. Die nicht mit der Umgebung verbundenen Neuronen bezeichnet man als „versteckte“ oder innere Neuronen.

In diesem Modell wird der Zeitablauf in diskrete Zeitpunkte unterteilt. Ein Signal gelangt im Allgemeinen von einem Neuron zu einem anderen. Eine **Propagierungsfunktion**

$input_j$  verarbeitet alle eingehenden Signale  $x_j$  von anderen Neuronen  $i$  zur Eingabe eines Neurons  $j$ . Typischerweise, so auch in den hier durchgeführten Experimenten, verwendet man die mit den Verbindungsgewichten  $w_{ij}$  gewichtete Summe. Also:

$$input_j = \sum_{i=1}^n x_j \cdot w_{ij} \quad (2.2)$$

wobei  $n$  die Anzahl der Neuronen des neuronalen Netzes darstellt.

Die **Aktivierungsfunktion** (auch: Transferfunktion) verarbeitet diese Eingabe und den aktuellen Aktivierungszustand des Neurons zum neuen Aktivierungszustand. Als Schwellenwert bezeichnet man den Wert des größten Anstiegs der Aktivierungsfunktion eines Neurons. Anschaulich beschrieben ist der Schwellenwert derjenige Wert, ab dem ein Neuron den Zustand „aktiv“ besitzt. Dieser ist in den folgenden Experimenten unterschiedlich und kann durch ein Schwellenwertneuron (auch: „Bias-Neuron“) eingestellt werden. Dieses liefert stets eins als Ausgabewert. Die Aktivierungsfunktion ist in den folgenden Experimenten stets die gleiche. Gängige Aktivierungsfunktionen sind Stufenfunktionen, lineare oder sigmoide<sup>1</sup> Funktionen. In der vorliegenden Implementierung lassen sich die Funktionen durch Ein- bzw. Auskommentieren ändern. Für die Experimente wird die sigmoide Aktivierungsfunktion  $\tanh$  (vgl. Abbildung 2.2) verwendet, also

$$activation_j = \tanh(input_j) \quad (2.3)$$

Die **Ausgabefunktion** berechnet aus dem aktuellen Aktivierungszustand eines Neurons dessen Ausgabewert. Häufig wird dafür die Identität  $id$  gewählt (so auch hier), also

$$output_j = id(activation_j) = activation_j \quad (2.4)$$

Die Architektur eines neuronalen Netzes ist festgelegt durch die Anzahl der Neuronen sowie die Verbindungen zwischen diesen. Die einzelnen Einheiten sind oft zu Schichten

---

<sup>1</sup>Funktionen mit s-förmigem Verlauf

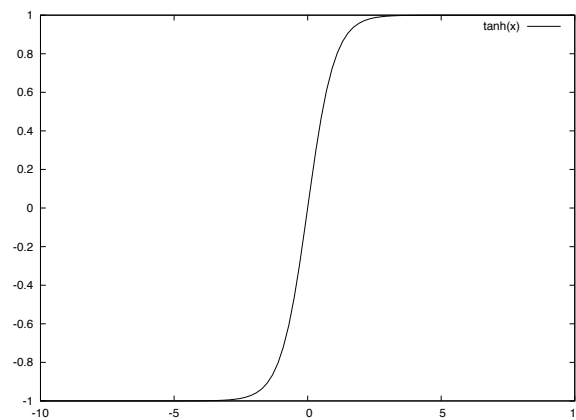


Abbildung 2.2: Der Tangens hyperbolicus als Aktivierungsfunktion des neuronalen Netzes

zusammengefasst (z.B. Eingabeschicht, innere Schicht und Ausgabeschicht). Allerdings herrscht in der Fachliteratur Uneinigkeit darüber, ob eine Schicht nun die Schicht der Neuronen oder die der Synapsen beschreibt. Solange jedoch klar ist, ob man sich auf Neuronen-Schichten oder Synapsen-Schichten bezieht, sollten keine Probleme auftreten.

Das neuronale Netz lässt sich bei gegebener Topologie durch eine Gewichtsmatrix beschreiben. Konkret gibt die Komponente  $a_{ij}$  in der hier gewählten Darstellung das Gewicht der Verbindung von Neuron  $j$  zu Neuron  $i$  an. Besteht kein Gewicht, so ist keine Verbindung vorhanden. Eine Verbindung mit einem Gewichtswert von null wird davon in der Regel unterschieden. Zur grafischen Veranschaulichung werden in der Matrix die Gewichtswerte nicht durch Angabe von Zahlenwerten, sondern über die Farbe der Einträge visualisiert. Diese Darstellung bezeichnet man auch als Hinton-Darstellung.

Damit ergibt sich eine quadratische Matrix, in der jedes Neuron durch genau eine Spalte und genau eine Zeile dargestellt wird.  $N$  beschreibe ein neuronales Netz mit  $\#e$  Eingabeneuronen, einem Schwellenwert-Neuron,  $\#v$  inneren Neuronen und  $\#a$  Ausgabeneuronen. Die ersten  $\#e + 1$  Zeilen der Matrix haben kleine weiße Kästchen. Da es sich hierbei entweder um Eingabeneuronen handelt, deren Eingabewerte von Sensoren stammen und die unverfälscht zur Verfügung stehen sollen, oder um ein Bias-Neuron, das keinen Eingabewert

bewert erhält und konstant den Ausgabewert eins liefert, können keine veränderbaren Synapsen von anderen zu diesen Neuronen existieren. Die ersten  $\#e + 1$  Spalten der Matrix sind, sofern es sich um Synapsen von Eingabeneuronen zu anderen Neuronen des neuronalen Netzes handelt, durch große weiße Kästen dargestellt. Schwarze Quadrate symbolisieren Synapsen von inneren Neuronen zu nachfolgenden Neuronen. Falls das schwarze Quadrat nicht ausgefüllt ist, handelt es sich um potentielle rekurrente Synapsen zu Vorgängerneuronen (oder zu dem eigenen Neuron). Farbig ausgefüllte Quadrate stellen tatsächlich vorhandene Synapsen dar. Eine beispielhafte Darstellung eines in den Versuchen evolvierten neuronalen Netzes findet sich in Abbildung 2.3.

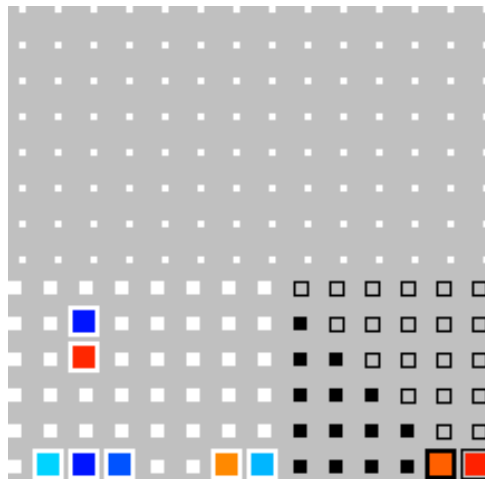


Abbildung 2.3: Beispielhaftes evolviertes Netz in der Hinton-Darstellung. Das Netz besitzt keine näher spezifizierte Funktionalität und dient nur der Veranschaulichung des Darstellungsprozesses eines neuronalen Netzwerks. Der Quellcode zur Erzeugung einer derartigen Darstellung stammt von [Mü10].

Ein neuronales Netz lässt sich auch als Adjazenzliste darstellen. Zu jedem Knoten wird dort eine Liste aller Nachbarn geführt. In dieser Liste ist zuerst das Schwellenwert-Neuron gespeichert, danach folgen Eingabeneuronen, innere Neuronen und abschließend Ausgabeneuronen. Wird im Folgenden eine Reihenfolge der Neuronen unterstellt, so bezieht sich



Damit neuronale Netze eingesetzt werden können, müssen sie „lernen“, d.h. die Synapsengewichte müssen (wiederholt) den Anforderungen angepasst werden. Lernen lässt sich im Groben auf die zwei Bereiche „Lernen durch Evolution“ („phylogenetic learning“) und „Lernen im Laufe des Lebenszyklus“ („phylogenetic learning“) aufteilen [AB97]. In den hier vorliegenden Experimenten wird nur die evolutionäre Form verwendet. Während ihres Lebenszyklus sind die Roboter zwar in der Lage, sich zu verändern. Allerdings können sie nicht beurteilen, ob diese Veränderung nun vorteilhaft ist oder nicht. Daher kann von Lernen während des Lebenszyklus im eigentlichen Sinne nicht die Rede sein. Die Beurteilung (und somit der eigentliche Lernprozess) findet erst durch den evolutionären Ansatz der impliziten Fitness statt. Anfänglich werden die Synapsengewichte in der Regel auf null gesetzt (sofern die Topologie bereits feststeht und nur trainiert werden soll), bevor sie beginnen, durch das verwendete Lernverfahren ihre Funktionalität zu entfalten.

Eine weitere Kategorisierung von Lernverfahren ist in Tabelle 2.4 durchgeführt. Hier wird eine Art unüberwachtes Lernen benutzt, d.h. den neuronalen Netzen werden nur die Eingabewerte präsentiert. Diejenigen, welche die gestellte Aufgabe nicht gut genug erfüllen, werden durch das evolutionäre Verfahren aus der Menge der Kandidaten entfernt.

### 2.2.2 Verwendung in der Evolutionären Robotik

Neuronale Netze haben sich in der Evolutionären Robotik als Controller, also Steuerprogramme, etabliert. Dazu haben einige bemerkenswerte Eigenschaften geführt. Beispielsweise sind (rekurrente) neuronale Netze turing-vollständig [Hyo96]. Darüber hinaus haben sie sich in Experimenten als relativ robust gegenüber Rauschen und ähnlichen Störungen erwiesen. Diese Fehlertoleranz ist besonders dann wünschenswert, wenn Steuerprogramme aus der Simulation in der Realität erprobt werden sollen. Außerdem führen geringe Änderungen am neuronalen Netz meist auch nur zu geringen Änderungen in der Verhaltensweise des dadurch gesteuerten Roboters. Des Weiteren ist das Modell eines neuronalen

Tabelle 2.4: Lernverfahren in neuronalen Netzen nach [NF01, Seite 27-44].

Typ	Charakteristika
Unüberwachtes Lernen	nur Eingabemuster gegeben
Bestärkendes Lernen	Eingabemuster und Rückmeldung gegeben, ob die Lösung gut oder schlecht war (ggfs. auch noch wie gut oder wie schlecht)
Überwachtes Lernen	Eingabemuster und optimale Lösung gegeben, d.h. der Fehlervektor steht dem neuronalen Netz zur Verfügung

Netzes ein biologisch plausibler Mechanismus für die Steuerung eines Individuums.

Der wahrscheinlich größte Nachteil von neuronalen Netzen liegt in ihrem sogenannten Blackbox-Verhalten. Bei einfachsten Netzen mag es für den Menschen noch möglich sein, deren Funktionsweise nachzuvollziehen. Steigt jedoch die Größe des Netzes und wird die Topologie komplexer, so ist es im Allgemeinen nicht mehr möglich, das Verhalten in einer bestimmten Situation bei gegebenen Sensorwerten ohne Weiteres zu bestimmen. Der Transfer von in der Simulation evolvierten neuronalen Netzen in die Realität gestaltet sich ebenfalls schwierig. König und Schmeck haben daher in ihren Arbeiten [KS08] [KMS09] deterministische Endliche Automaten als Steuerprogramme verwendet. Der besondere Reiz von Endlichen Automaten liegt sicherlich in der einfachen Nachvollziehbarkeit der Verhaltensweisen. Es lässt sich bei einem Endlichen Automaten garantieren, dass er sich bei gegebenen Sensorwerten in einer bestimmten Situation in einer vorhersagbaren Art und

Weise verhält. Endliche Automaten sind zwar nicht turing-vollständig, allerdings zeigen die Untersuchungen, dass sich zumindest grundlegende Verhaltensweisen damit evolvieren lassen.

Wenn mehrere Roboter allerdings in einem Schwarm agieren, so lässt sich das emergierende Verhalten nicht allein durch eine isolierte Betrachtung der Individuen bestimmen. Dadurch, dass einzelne Roboter miteinander und der Umwelt in Interaktion stehen, lassen sich zwar deutlich komplexere Problemstellungen lösen; eine unmittelbare Vorhersage des Verhaltens (auf Schwarmebene) ist aber selbst bei Endlichen Automaten nicht mehr möglich. Nach sorgfältiger Abwägung von Vor- und Nachteilen wurden die hier Experimente mit Individuen durchgeführt, die neuronalen Netze als Steuerprogramme verwenden.

# Kapitel 3

## Implementierung

Das Implementierungskapitel beginnt mit einer Einordnung des implementierten Modells in bereits bestehende ähnliche Modelle und beschreibt den evolutionären Operator, der den Versuchen zu Grunde liegt. Anschließend wird die Simulationsumgebung vorgestellt, bevor näher auf das eigentliche Szenario eingegangen wird. Danach wird die implementierte Sensorik und Aktorik von Räuber und Beute beschrieben. Abschließend werden weitere Implementierungen präsentiert, die für die Durchführung der Versuche vorzunehmen waren.

### 3.1 Einordnung des implementierten Modells

Das implementierte Modell hat Ähnlichkeiten zu „NeuroEvolution of Augmenting Topologies“ (NEAT) [SM02] und zu Cascade Correlation [FL90] bzw. Cascade Correlation in der erweiterten rekurrenten Version [Fah90]. Daher werden diese Herangehensweisen im Folgenden kurz näher beschrieben.

Bei vielen Ansätzen wird zunächst eine feste Struktur des neuronalen Netzes gewählt und dessen Verbindungsgewichte optimiert. Diese besteht gewöhnlich aus einer Schicht von inneren Neuronen, von denen jedes einzelne mit allen Eingabe- sowie allen Ausga-

beneuronen verbunden ist. Durch die Evolution wird der Ergebnisraum durchsucht und die Gewichte zwischen den einzelnen Neuronen werden optimiert. Die vorher festgelegte Struktur des neuronalen Netzwerks bleibt jedoch weiterhin fest — unabhängig davon, ob diese überhaupt die optimale Lösung darstellt.

NEAT geht einen anderen Weg. Neben der Topologie beschreibt es auch ein Lernverfahren für neuronale Netze. Dabei optimiert es simultan sowohl Gewichte als auch Struktur des Netzes und ist dadurch Ansätzen mit fest vorgegebenen Netzgrößen überlegen. Es lässt prinzipiell beliebige Verbindungen zwischen den Neuronen zu. Ein neu eingefügtes Neuron wird grundsätzlich verbunden eingefügt und ersetzt eine Synapse, wobei das neue Neuron mit den zwei Endpunkten der entfernten Synapse verbunden wird. Damit soll die bestehende Funktion erhalten und erweitert werden. Dies führt ein Stück weit allerdings auch zur Erschwerung von Innovation, gerade weil die bestehende Funktion erhalten wird. Die einzelnen Synapsen sind zusätzlich durchnummeriert. Diese Nummern werden sowohl für einen Kreuzungsoperator als auch im Zusammenhang mit einem Mechanismus verwendet, der durch eine Art Nischenbildung Innovationen schützen soll.

Cascade Correlation lässt ebenfalls beliebige Verbindungen zwischen Neuronen zu. Dieses Verfahren beschreibt neben einer Topologie auch einen Lernalgorithmus. Gestartet wird mit einem Netz geringer Größe, bevor Schritt für Schritt jeweils ein neues Neuron hinzugefügt wird und anschließend die Verbindungsgewichte trainiert werden. Nachdem ein Neuron hinzugefügt worden ist, werden die Eingabewerte des Netzes eingefroren. Danach wird das Netz mittels einer Form des überwachten Lernens (vgl. Tabelle 2.4) trainiert. Das neue Neuron kann eine bestimmte Funktion übernehmen sowie seinerseits wieder einen Eingabewert für später eingefügte Neuronen oder einen Ausgabewert des neuronalen Netzes bereitstellen. Dadurch ergibt sich Stück für Stück eine mehrschichtige Struktur. In Experimenten hat sich gezeigt, dass die Vorteile dieses Verfahrens schnelles Lernen, eine selbstständige Größen- und Topologiebestimmung sowie ein Beibehalten der Struktur bei

Änderung der Trainingsmenge sind.

In den Experimenten werden die Beutetiere durch neuronale Netze als Steuerprogramme gesteuert. Im Laufe des evolutionären Prozesses ist sowohl eine Änderung der Topologie der künstlichen neuronalen Netze als auch ein Modifizieren der Verbindungsgewichte möglich. Ausgehend von einer bestimmten Netzgröße (hier: innere Neuronenanzahl gleich drei) wird dieses im Laufe der Evolution vergrößert bzw. verkleinert. Anfangs werden die Netze 1000 mal mutiert, um mit zufälligen und nicht mit komplett leeren neuronalen Netzen zu starten.

Die Lernverfahren aus NEAT und Cascade Correlation werden in den folgenden Experimenten nicht verwendet. Stattdessen wird ein evolutionärer Algorithmus mit folgenden genetischen Operatoren eingesetzt:

- **Neuron einfügen:** Fügt ein Neuron an beliebiger Stelle zwischen Ein- und Ausgabeneuronen ein. Die Wahrscheinlichkeit einer derartigen Operation ist geringfügig niedriger als die von Löschooperationen, damit das Netz nicht übermäßig wächst.
- **Synapse einfügen:** Fügt eine Verbindung zwischen zwei Neuronen ein (nur möglich, wenn noch keine Verbindung besteht). Die Wahrscheinlichkeit einer derartigen Operation ist geringfügig niedriger als die von Löschooperationen.
- **Neuron löschen:** Löscht ein Neuron. Verbindungen von und zu diesem Neuron werden ebenfalls entfernt. Die Wahrscheinlichkeit einer derartigen Operation ist geringfügig höher als die von Einfügeoperationen.
- **Synapse löschen:** Löscht eine Synapse zwischen zwei Neuronen. Die Wahrscheinlichkeit einer derartigen Operation ist geringfügig höher als die von Einfügeoperationen.
- **Gewicht verändern:** Verändert das Verbindungsgewicht durch Addition des Werts einer standardnormalverteilten<sup>1</sup> Zufallsvariable auf das bestehende Verbindungsge-

---

<sup>1</sup> $\mathcal{N}(\mu, \sigma)$ -verteilten Zufallsvariable mit  $\mu = 0$  und  $\sigma = 1$

wicht. Die neuen Synapsengewichte  $w_{ij}$  zum Zeitpunkt  $t + 1$  ergeben sich also zu

$$w_{ij}(t + 1) = w_{ij}(t) + \delta \quad (3.1)$$

wobei  $\delta$  gerade eine Realisation der obigen Zufallsvariable darstellt. Die Wahrscheinlichkeit einer derartigen Operation ist geringfügig höher als die von Einfügeoperationen.

## 3.2 Simulationsumgebung

Erste Experimente wurden mit dem von Lukas König entwickelten Java-Framework Finite Moore Generator (FMG) durchgeführt. Zur Steuerung der Roboter werden dort endliche Moore Automaten verwendet. Mit entsprechendem Programmierungsaufwand ist es auch dort möglich, neuronale Netze als Controller zu verwenden, obwohl das Programm auf Endliche Automaten ausgelegt ist. Bei dem in FMG simulierten Robotertyp handelt es sich um den Jasmine IIIp Roboter, einem nur ca.  $26 \times 26 \times 26 \text{ mm}^3$  großen Roboter, der Infrarotsensoren besitzt, um Entfernungen zu messen und um mit anderen Robotern zu kommunizieren. Daneben kann der Mikro-Roboter vorwärts und rückwärts fahren und sich nach links oder rechts drehen.

Die Experimente, die in dieser Arbeit betrachtet werden, stammen ausnahmslos aus der Weiterentwicklung von FMG, genannt Easy Agent Simulation (EAS). Hier ist der Benutzer nicht mehr auf eine bestimmte Controllerart festgelegt, was die Benutzung von neuronalen Netzen als Steuerprogramm wesentlich vereinfacht. Generell wird dem Anwender in dieser Version größtmögliche Freiheit gelassen. So gibt es zum Beispiel keinen bestimmten simulierten Robotertyp mehr wie im Fall von FMG. Allerdings modelliert man in Simulationen in der Regel reale Roboter, denn schließlich sollen die Ergebnisse über kurz oder lang in die Realität übertragen werden. Easy Agent Simulation erlaubt das Design beliebig ausgestatteter Roboter und ist daher flexibel und anpassungsfähig. Auf der anderen Seite ist allerdings zunächst eine vollständige Beschreibung des zu si-

mulierenden Roboters nötig. Der Benutzer entscheidet selbst über die komplette Sensorik und Aktorik aller Roboter, deren Verhalten in der Simulation studiert werden soll. Das führt zum einen zwar zu einem deutlich erweiterten Funktionsspektrum, zum anderen muss der Benutzer sich jedoch selbst Gedanken über den fiktiven Roboter sowie dessen Interaktionsmöglichkeiten mit der Umwelt machen und (Standard-)Sensoren implementieren. Bedingt durch den Umstieg von FMG zu EAS handelt es sich allerdings noch um eine frühe Version, zukünftig werden wohl Standardsensoren u.ä. bereits verfügbar sein.

Die Simulation läuft in dieser Version in Zyklen (auch: „ticks“) ab. In jedem Zyklus werden Sensorwerte generiert, die nächste Aktion berechnet und anschließend ausgeführt. Nach bestimmten Ereignissen (Verstreichen eines vorab definierten Zeitfensters) werden schließlich Evolutionsoperatoren genutzt.

Es hat sich im Laufe der Experimente herausgestellt, dass die Kollisionsabfrage einer der zeitintensivsten Vorgänge zu sein scheint. Dies kommt hier insbesondere zum Tragen, da über 100 Agenten auf dem Spielfeld aktiv sind. Um einen Performance-Zuwachs zu erreichen, wurde daher zunächst in Betracht gezogen, die Kollisionsabfrage abzustellen, da diese für die untersuchte Fragestellung von untergeordneter Bedeutung ist. Es stellte sich im Folgenden jedoch heraus, dass die Agenten (Schafe), deren Abfrage abgeschaltet wurde, das Spielfeld verlassen haben (und damit verhindert haben, von den Wölfen erbeutet zu werden). Das Spielfeld wird bei EAS im Gegensatz zu FMG nämlich nicht mehr durch eine importierte Grafikdatei, sondern durch spezielle Wand-Agenten repräsentiert (vgl. Abbildung 3.1). Daher kann bei Abschalten der Kollisionsabfrage ein Verlassen des Spielfelds durch einzelne Agenten nicht mehr unterbunden werden. Diese Idee zu Geschwindigkeitsoptimierung wurde daher wieder verworfen.

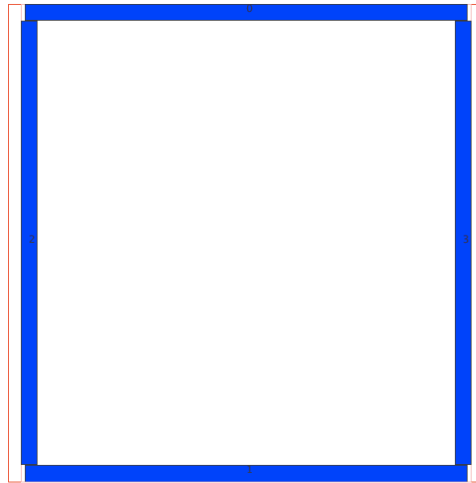


Abbildung 3.1: Visualisierung der Umgebung, in welcher die Agenten platziert werden. Die Wände der Simulation sind aus vier einzelnen Wand-Agenten zusammengesetzt.

### 3.3 Szenario

Das Räuber-Beute-Szenario an sich ist häufig bei Experimenten zur Evolutionären Robotik anzutreffen. Räuber (beispielsweise Wölfe oder Geparden) müssen dabei Nahrung in Form von Beutetieren (beispielsweise Schafe oder Gazellen) jagen, um ihren Fortbestand zu sichern. Letztere müssen versuchen, den Jägern auszuweichen bzw. zu entkommen, um ihr eigenes Überleben zu sichern.

Dazu werden neben den üblichen Distanzsensoren noch einige andere Sensoren implementiert, z.B. um die Entfernung sowie die Richtung des nächsten Raub- bzw. Beutetiers wahrzunehmen, was eine der Situation entsprechende Schwarmbildung ermöglichen soll. Wie in der Natur ist das Raubtier (insbesondere auf kurzen Strecken) schneller als das Beutetier und zeichnet sich durch besonders scharfe Sinne aus (vgl. Räuber-Beute-Beziehung zwischen Gepard und Gazelle, Wolf und Schaf usw.).

In der Tierwelt stellt sich das skizzierte Szenario wie folgt dar: Je mehr Beutetiere existieren, desto mehr Räuber können am Leben bleiben, weil sie genügend Futter finden und erbeuten können. Eine Erhöhung der Beutepopulation geht ceteris paribus mit einer

zeitversetzten Erhöhung der Raubtierpopulation einher, die ihrerseits wieder dazu führt, dass die Anzahl der Beutetiere abnimmt. Dies führt zu einer Entwicklung eines biologischen Gleichgewichts, welches das Wachstum der beiden Populationen in Grenzen hält. Diesen Aspekt, bekannt als das erste Lotka-Volterra-Gesetz (periodische Schwankungen der Populationsgrößen), und weitere Erkenntnisse haben Alfred James Lotka und Vito Volterra in den Lotka-Volterra-Gesetzen festgehalten. Dort werden einige quantitative Zusammenhänge bei Räuber-Beute-Beziehungen näher beschrieben, wie die Feststellung konstanter Mittelwerte der Populationsgrößen (Zweites Lotka-Volterra-Gesetz) und ein schnelleres Erholen der Individuenanzahl bei Störung der Räuber-Beute-Beziehung (Drittes Lotka-Volterra-Gesetz). Von diesen Gesetzen wird jedoch in Experimenten im Kontext Evolutionärer Robotik zum Teil abstrahiert. In den durchgeführten Versuchsläufen ist es beispielsweise so, dass die Raubtiere nicht verhungern können und somit deren Populationsgröße während des gesamten Experiments konstant ist.

Die relative Anzahl der Räuber zur Anzahl der Beute beträgt konstant eins zu fünf, um zu verhindern, dass die Beutepopulation vollständig ausgelöscht wird. Ein entsprechendes Verhältnis aus der Natur abzuleiten gestaltet sich schwierig. Das Problem, auf welches man zwingendermaßen stößt, ist, dass die Anzahl von erbeuteten Individuen keinen Rückschluss auf die daraus tatsächlich resultierende Futtermenge zulässt. So frisst ein Wal täglich mehrere Tonnen Krill, Kleinkrebse mit einem Gewicht von wenigen Gramm, wohingegen beispielsweise die Anzahl der von Geparden erbeuteten Gazellen im unteren einstelligen Bereich liegt. Ein festes Räuber-Beute-Verhältnis rein zahlenmäßig festzulegen ist somit im Allgemeinen problematisch bzw. nicht möglich.

Im klassischen Räuber-Beute-Szenario der Evolutionären Robotik wird eine Zusammenarbeit der Individuen innerhalb einer Gruppe — wenn überhaupt — nur in der Gruppe der Räuber untersucht (vgl. Kapitel 1). Dabei orientiert man sich an der in der Natur ebenfalls anzutreffenden Rudelbildung, um die Erfolgsaussichten einer Jagd zu erhöhen. Im hier un-

tersuchten Szenario wird Kooperation innerhalb der Beutepopulation untersucht. Es ist im Hinblick auf die Erfüllung der Aufgabenstellung vorteilhaft, mit den eigenen Gruppenmitgliedern zu kooperieren. Jedes Beutetier an sich ist den Raubtieren hinsichtlich seiner Fortbewegungsgeschwindigkeit unterlegen. Erstere haben genau dann eine höhere Überlebenschance, wenn sich zu Verbänden von mehreren Tieren zusammenschließen. Das soll — so die Idee — dazu führen, dass der Räuber durch die große Anzahl an Beutetieren „verwirrt“ wird und bei der Jagd zunehmend zwischen den einzelnen Beutetieren hin- und herwechselt. Wenn diese Verwirrungstaktik ein gewisses Ausmaß erreicht hat, wird dem Räuber die Jagd deutlich erschwert. Die Beute hat in diesem Fall durch Kooperation innerhalb der eigenen Gruppe ihre eigenen Überlebenschancen erhöht.

Ein typischer Simulationslauf läuft dabei folgendermaßen ab: Zunächst werden die Roboter zufällig auf dem Spielfeld verteilt. Anschließend startet der eigentliche Evolutionslauf. Alle Individuen verfolgen ihre Strategie bzw. müssen diese zunächst erst evolvieren, wobei die Fitness implizit evaluiert wird. Nach  $\alpha$  ticks werden die Netze jeweils mutiert ( $\alpha$  ist durch den Parameter „brainMutationInterval“ definiert). Die Länge des gesamten Laufs beträgt stets 300.000 Zyklen.

## 3.4 Roboter

### 3.4.1 Räuber

Die Breite der Räuberrepräsentation (vgl. Abbildung 3.2) in der Simulation dient im Folgenden als Standardmaß, zu dem alle anderen Längenmaße dieser Arbeit relativ dazu angegeben werden (analog zum Konzept eines Numeraire-Gutes in der Volkswirtschaftslehre). Damit entspricht diese Breite einer Wolf-Längeneinheit (WLE). Andere Längenmaße dieser Simulation werden als Vielfache dieser Breite angegeben.

In den hier durchgeführten Experimenten ist die Jagdstrategie der Räuber fest implemen-

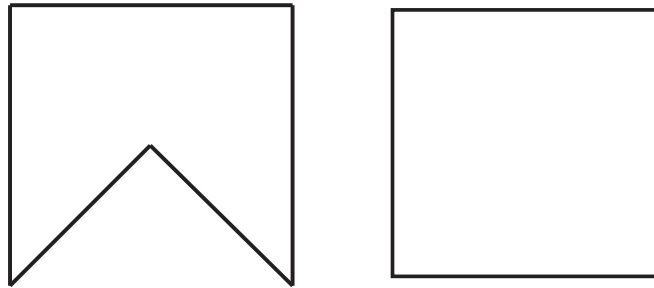


Abbildung 3.2: Visualisierung der Agenten in der Simulation. Links ein Raubtier, rechts ein Beutetier. Die Länge und die Breite der beiden Agenten beträgt je 1 WLE. Agenten werden in der Simulation anhand ihrer Identifikationsnummer (ID) unterschieden:  $ID < 10$ : Wand-Agenten,  $ID \geq 10$  und  $ID \bmod 2 = 0$ : Schaf-Agenten,  $ID \geq 10$  und  $ID \bmod 2 = 1$ : Wolf-Agenten.

tiert und von der Evolution ausgenommen. Bei dieser Strategie wird alle  $\beta$  Zyklen (wobei  $\beta$  für den Wert des Parameters `chooseSheepInterval` steht) überprüft, ob sich ein Beutetier in einem zuvor durch einen weiteren Parameter  $\gamma$  (`sheepDetectionRadius`) festgelegten Umkreis befindet. Wenn dies der Fall ist, dann verfolgt der Räuber das Beutetier so lange, bis er es entweder erbeutet hat oder bis ein neues Beutetier ausgewählt wird. Wenn eine Kollision mit einem anderen Agenten, also der Wand oder einem anderen Raubtier, droht, bekommt der Roboter zufällig so lange einen neuen Blickwinkel zugewiesen (in dessen Richtung er sich dann weiter bewegt), bis die entsprechenden Distanzsensoren wieder unkritische Werte liefern. Diese Vorgehensweise ist in Algorithmus A1 zusammengefasst.

**Algorithmus A1**

(\*Strategie der Wölfe\*)

**Input:** Beute-Erkennungsradius (*sheepDetectionRadius*),Beuteauswahlintervall (*chooseSheepInterval*),

alle verfügbaren Sensorwerte

**Output:** Verhalten des Wolfs (definiert durch ein bestimmtes Zusammenspiel der Aktoren)**if** Schaf in Beute-Erkennungsradius (*sheepDetectionRadius*)    **if** seit länger als Beuteauswahlintervall (*chooseSheepInterval*) kein neues Ziel gewählt        wähle neues Ziel (*chooseSheep*)    **end if**    fokussiere Ziel (*focus*)**end if****if** (durch *sensing - agent - x*,  $x = 1 \dots 3$  wahrgenommener Agent/ID ist Wand (d.h.  $ID < 10$ )oder anderer Wolf ( $ID \bmod 2 = 1$ )) und Entfernung zum wahrgenommenenAgenten (Wand oder Wolf) ist  $<$  kritischer Grenzwert

(\*verhindere Zusammenstoß\*)

    infinitesimal rückwärts fahren (*avoid - collision*)    Blickwinkel zufällig neu setzen (*change - angle*)**end if****else**    fahre vorwärts (*drive*)**end else**fange Schaf, wenn möglich (*catchSheepIfPossible*)

## Sensoren

- **sensing-agent-x**,  $x=1\dots 3$  ( $-45^\circ$ ,  $0^\circ$ ,  $+45^\circ$  in Blickrichtung). Dient zum Erkennen der Art (d.h. Wand, anderes Raubtier oder Beutetier) des bevorstehenden Hindernisses (vgl. Abbildung 3.3).
- **distance-x**,  $x=0\dots \delta$ : Distanzsensoren des Raubtieres, festgelegt durch Parameter  $\delta$ , hier:  $\delta$  konstant gleich 3 (vgl. Abbildung 3.3).
- **sheepInDetectionRadius**: Liefert wahr („true“) zurück, wenn ein Schaf im Beutetier-Erkennungsradius ist („Revier des Räubers“).
- **targetPositionInDetectionRadius**: Stellt dem Räuber die Position des Beutetieres zur Verfügung, sofern sich das Beutetier im Erkennungsradius befindet.
- **targetID**: Beinhaltet die ID des gerade verfolgten Beutetiers. ID ist eine simulationsweit eindeutige Nummer eines jeden Individuums.

## Aktoren

- **drive**: Dient zum Fortbewegen in Blickrichtung.
- **chooseSheep**: Wählt ein zufälliges Beutetier aus, das sich im Beutetier-Erkennungsradius des Raubtieres befindet.
- **focus**: Fokussiert ein Beutetier, d.h. richtet die Blickrichtung in Richtung des Beutetiers.
- **avoid-collision**: Dient zum Rückwärtsbewegen, falls eine Kollision mit einem anderen Agenten droht.
- **change-angle**: Setzt eine zufällige Blickrichtung (Teil der Kollisionvermeidungsstrategie).

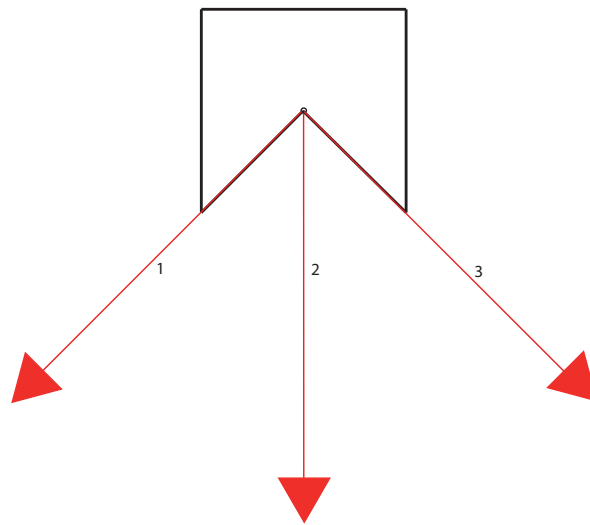


Abbildung 3.3: Distanz- und Erkennungssensoren der Wolf-Roboter. Die roten Pfeile geben die Richtung der Sensoren sensing-agent- $x$ ,  $x=1...3$  und distance- $x$ ,  $x=1...3$  an. Die Blickrichtung des Wolfs entspricht der Richtung des roten Pfeils mit der Nummer 2

- **catchSheepIfPossible:** Akteur, um ein Beutetier zu fangen, das sich im Fangradius des Raubtieres befindet.

### 3.4.2 Beute

Die Beutetiere, deren Verhalten durch neuronale Netze gesteuert wird, sind dem evolutionären Prozess unterworfen. Anfangs bestehen die neuronalen Netze der Schafe (vgl. Abbildung 3.4) aus einem Schwellenwertneuron, Eingabeneuronen, drei inneren Neuronen und zwei Ausgabeneuronen, welche die Geschwindigkeit der Räder (links und rechts) des Roboters steuern. Die Anzahl der Eingabeneuronen entspricht gerade der Anzahl an Sensoren. Alle Sensorwerte aus der Umgebung werden dem künstlichen neuronalen Netz als Eingabeneuronen zur Verfügung gestellt. Die Ausgabeneuronen des Netzes steuern das linke und rechte Rad des simulierten Roboters. Die Geschwindigkeit des linken bzw. rechten Rads bestimmt sich nach der folgenden Vorschrift:

$$v_{links} = \min\{0.5 + outs[0] \cdot v_{max}, v_{max}\}$$

$$v_{right} = \min\{0.5 + outs[1] \cdot v_{max}, v_{max}\}$$

wobei  $outs[0]$  dem (Ausgabe-)Wert des ersten,  $outs[1]$  dem Wert des zweiten Ausgabeneurons entspricht.

Diese Berechnung stellt sicher, dass sich die Beutetiere bereits zu Beginn bewegen, damit höhere Geschwindigkeiten favorisiert und die Maximalgeschwindigkeit der Beuteroboter nicht überschritten werden. Fähigkeiten wie Umherfahren oder Ausweichen sind für die spätere Entwicklung von komplexeren Verhaltensweisen unabdingbar, denn ein gejagtes Tier, das stehen bleibt, ist eine leichte Beute für den Räuber.

Wenn ein Schaf gefangen wird, so wird der Beuteroboter vom Spielfeld entfernt. Das neuronale Netz wird anschließend gelöscht, bevor zufällig aus den noch im Pool, d.h. in der aktuellen Schaf-Population befindlichen neuronalen Netzen, eines ausgewählt wird. Schließlich wird dieser mit einem neuen Netz versehene Roboter von neuem zufällig auf dem Spielfeld platziert.

Die Anzahl der Distanzsensoren bestimmt sich durch den Parameter *numberOfDistance Sensors*. Je nach Wahl werden diese Distanzsensoren symmetrisch um den Roboter herum verteilt. In ersten Experimenten konnten diese Sensoren nur Roboter erkennen, falls der Strahl einen anderen Roboter traf. Später wurden die Sensoren dahingehend angepasst, dass sie alle Roboter in den einzelnen Kreissektoren wahrnehmen konnten. Für *numberOfDistance Sensors* = 6 Sensoren ergibt sich die Situation wie in Abbildung 3.5 dargestellt.

### Sensoren

- **distance-x, x=0... $\delta$** : Distanzsensoren des Raubtieres, festgelegt durch Parameter  $\delta$ .
- **sensing-wolf-distance**: Gibt Entfernung zum nächsten Raubtier an.

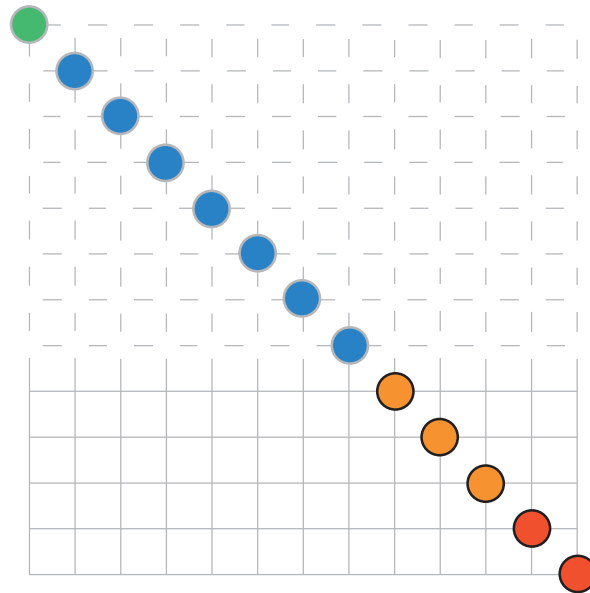


Abbildung 3.4: Visualisierung des anfänglichen neuronalen Netzes. Von links oben beginnend nach rechts unten zuerst das Schwellenwertneuron (grün), das keine Eingänge besitzt und als Ausgabe konstant eins liefert. Danach in blau die Eingabeneuronen; deren Anzahl entspricht exakt der Anzahl der Beutetier-Sensoren (das dargestellte Netz gilt für genau drei Distanzsensoren, Experimente wurden auch mit sechs Distanzsensoren durchgeführt, dazu sind dann drei weitere Eingabeneuronen der Darstellung hinzuzufügen). Anschließend in orange die inneren Neuronen (anfangs drei Stück) sowie als letztes in rot die Ausgabeneuronen. Im implementierten Modell können Synapsen nur an den Gitterkreuzungspunkten entstehen. Vorwärtsgerichtete Synapsen sind links von der Diagonalen zu finden, die von links oben nach rechts unten führt. Synapsen rechts von dieser Diagonale kennzeichnen rückwärtsgerichtete (rekurrente) Synapsen. Ausgehend von diesem Grundgerüst werden die Netze vor erstmaliger Inbetriebnahme 1000 mal mutiert, um mit „zufälligen“ Netzen zu starten.

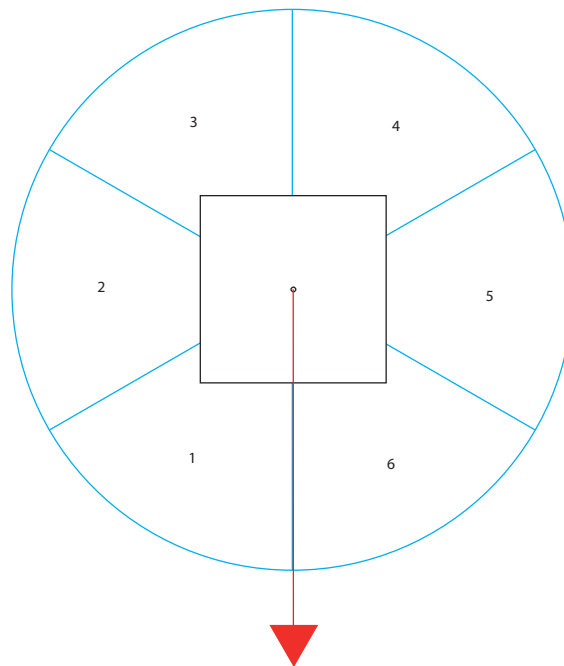


Abbildung 3.5: Die Distanzsensoren werden stets symmetrisch um die Roboter verteilt. Der rote Pfeil gibt die Blickrichtung des Roboters an. Die Abbildung gilt für sechs Distanzsensoren. Experimente wurden auch mit drei Distanzsensoren durchgeführt. Dafür sind jeweils zwei nebeneinanderliegende Bereiche zusammenzufassen.

- **sensing-wolf-direction:** Gibt Richtung des nächsten Raubtieres an.
- **sensing-sheep-distance:** Gibt Entfernung zum nächsten Beutetier an.
- **sensing-sheep-direction:** Gibt Richtung des nächsten Beutetieres an.

## Aktoren

- **drive:** Dient zum Fortbewegen in Blickrichtung.

## 3.5 Weitere Implementierungen

### 3.5.1 Statistik-Plugin

Um die einzelnen Versuchsläufe auszuwerten, musste zunächst ein Statistik-Plugin programmiert werden. Dabei werden die Koordinaten der Agenten, ihre jeweiligen Blickwinkel, die Anzahl der Beuteerfolge der Räuber sowie die Anzahl der Beutetiere, die sich in einem bestimmten Radius um jedes Beutetier befinden, aufgezeichnet. Der Parameter *recordingInterval* gibt an, in welchen Zeitabständen diese Werte in eine Comma-Separated-Values-Datei (CSV-Datei) geschrieben werden. Als Trennzeichen kommt allerdings nicht das Komma, sondern jeweils ein Semikolon zum Einsatz. Die Trajektorien werden darüber hinaus noch grafisch visualisiert und werden alle *saveTrajectoriesEach* Zyklen in eine PNG-Datei geschrieben. Die Pfade der Räuber sind dabei rot, die der Beute blau gekennzeichnet. Aufgrund der hohen Anzahl an Individuen in der Simulation lassen sich Trajektorien von einzelnen Agenten auf diesen Bildern nicht mehr nachvollziehen, es lassen sich aber durchaus qualitative Aussagen über das Verhalten der Gesamtpopulation treffen. Das Statistik-Modul berechnet zusätzlich in jedem Block bestehend aus *calculateStatisticsEachKTicks* Zyklen automatisch statistische Kennzahlen, welche ebenfalls in einer Datei zur Auswertung gespeichert werden (diese Kennzahlen werden

hier allerdings nicht verwendet). Diese Kennzahlen sind im Einzelnen das Minimum, das Maximum, das arithmetisches Mittel, der Median, die Varianz sowie die korrigierte Stichprobenstandardabweichung.

### 3.5.2 Export von neuronalen Netzen

Der Genotyp der im Laufe der Simulation evolvierten neuronalen Netze wird am Ende jedes Simulationslaufs abgespeichert. Dazu wird die in Java integrierte Objektserialisierung verwendet. Vorteile dieses Vorgehens sind die einfache Implementierung und die speicherplatzsparende binäre Abspeicherung des Netzes. Das auf diese Weise persistent gemachte Netz lässt sich bei Bedarf wieder deserialisieren und in andere Simulationsläufe einbinden.



# Kapitel 4

## Experimente

Das Experimente-Kapitel lässt sich in zwei Teile aufteilen: Zunächst werden die Parameter der Experimente festgelegt und Auswertungsstrategie präsentiert. Anschließend wird diese Evaluationsmethodik auf die Versuche angewendet, bevor auf einige Simulationsläufe genauer eingegangen wird. Die Versuche werden unter Zuhilfenahme des EAS-Frameworks auf bis zu 110 Rechnern der Fakultät für Wirtschaftswissenschaften des Karlsruher Instituts für Technologie parallel ausgeführt. Unter anderem bedingt durch die hohe Individuenanzahl (siehe unten) dauert ein Simulationslauf — je nach Leistung des verwendeten Computers — zwischen 11 und 26 Stunden.

### 4.1 Methodik der Evaluation

Insgesamt wurden 360 Simulationsläufe mit einer Länge von jeweils 300.000 Zyklen durchgeführt und ausgewertet. Da es sich um Simulationen handelt, die — bedingt durch das Szenario, das eine gewisse Anzahl von Raub- und Beutetieren erfordert — äußerst zeitaufwändig sind, wäre eine vollständige Untersuchung aller Parameter nicht möglich, ohne den Rahmen dieser Arbeit zu sprengen. Daher wurden vor allem diejenigen Parameter ausgewählt, die vielversprechend erschienen.

Die Simulationsläufe werden mit einer Individuenanzahl von 100 Schafen und 20 Wölfen durchgeführt. Weiterhin gibt es, bedingt durch die Funktionsweise von EAS, vier Wand-Agenten, die das Spielfeld begrenzen. Schafe haben in der Simulation eine *gerade* Simulations-ID größer oder gleich 10, Wölfe entsprechend eine *ungerade* Simulations-ID größer 10. Simulations-IDs kleiner als 10 sind für Wand-Agenten reserviert. Alle 10 Zyklen werden Momentaufnahmen der aktuell durchgeführten Läufe gespeichert. Aus den so gewonnenen Daten lassen sich die Läufe rekonstruieren und anschließend analysieren. Der herangezogene Standardparametersatz ist im Anhang aufgeführt.

Die Größe des Spielfeldes der Simulationsumgebung beträgt konstant ca. 500 mal 500 Wolf-Längeneinheiten (WLE). Der Anteil an Raubtieren an der Testpopulation liegt konstant bei 20 Prozent. Der Räuber-Beute-Erkennungsradius wird mit 50 WLE getestet. Der Räuber-Beute-Fangradius ist auf 5 WLE festgelegt. Das Verhältnis Räubergeschwindigkeit zu Beutegeschwindigkeit wird zwischen 1, 1.2 und 1.5 variiert. Die Nummer der Schaf-Distanzsensoren nimmt die Werte 3 und 6 an. Unabhängig davon haben die Wölfe stets drei Distanzsensoren. Das Beutetier-Auswahl-Intervall unterscheidet sich bei den einzelnen Läufen nicht, es liegt stets bei 20 Zyklen. Der Parameter für die anfängliche Gehirngröße ist auf 3 eingestellt. Da sich das neuronale Netz durch Mutation beliebig vergrößern und verkleinern kann, spielt dieser Parameter in den Experimenten nur eine untergeordnete Rolle. Das Mutationsintervall des neuronalen Netzes wird mit den Werten 50, 100 und 200 belegt. Ein weiterer Parameter, welcher die minimale Zeit angibt, ab der ein neuronales Netz an ein anderes Individuum weitergegeben werden kann, liegt bei 0.

Bevor im weiteren Verlauf des Kapitels auf die genauen Simulationsergebnisse eingegangen wird, wird zunächst der Auswertungsprozess genauer beschrieben. Basis der Auswertung sind die während der Simulation aufgezeichneten Werte und Kennzahlen. Anfangs werden möglichst viele Läufe herausgefiltert, die höchstwahrscheinlich kein vielversprechendes Verhalten aufweisen. Einen ersten Anhaltspunkt dafür liefern zum einen die An-

zahl der Beuteerfolge der Räuber und zum anderen die Anzahl der Beutetiere im Umkreis um jedes einzelne Beutetier. Diese werden am Ende der Simulation jeweils in einem separaten Schaubild dargestellt. Dabei werden die folgenden Daten in Abhängigkeit der Simulationszyklen dargestellt: der Absolutwert der gefangenen Schafe (bezogen auf ein 10 Zyklen umfassendes Intervall), der gleitende Durchschnitt der gefangenen Schafe über 25.000 Zyklen, das Minimum, das Maximum und der Durchschnitt der Anzahl von Schafen im definierten Radius (50 WLE) auf Basis der Population zum entsprechenden Zeitpunkt sowie der gleitende Durchschnitt ebenfalls über 25.000 Zyklen. Der gleitende Durchschnitt soll die starken Fluktuationen des einfachen Durchschnitts glätten, damit etwaige Trends der Kurven besser abgelesen werden können.

Wenn sich das erwünschte Verhalten entwickelt, dann sollte der gleitende Durchschnitt der gefangenen Schafe eine sinkende Tendenz, die Anzahl von Schafen im bestimmten Umkreis hingegen eine eher steigende Tendenz aufweisen. Geringe Schwankungen dieser beiden Kennzahlen sind aufgrund einer zufälligen Neu-Platzierung der Schafe auf dem Spielfeld zu erwarten. Ergibt sich keine Änderung des Levels dieser Kennzahlen, so muss davon ausgegangen werden, dass kein erwünschtes Verhalten evolviert wurde. Problematisch ist dabei allerdings, dass im Extremfall nur eine Teilmenge der Individuen in der Population das erwünschte Verhalten zeigen könnte, was dazu führen würde, dass sich der gleitende Durchschnitt nur wenig ändern und ein dementsprechendes Verhalten zunächst nicht erkannt werden würde. Des Weiteren werden die einzelnen Trajektorien auf systematische Verhaltensweisen wie Ansammlungen o.ä. untersucht. Durch das Zusammenspiel dieser Mechanismen soll die Wahrscheinlichkeit möglichst niedrig gehalten werden, gute Verhaltensweisen im Sinne der Aufgabenstellung zu übersehen. Die endgültige Bestätigung eines dauerhaft positiven Verhaltens erfolgt allerdings erst durch einen erneuten Test mit den evolvierten Netzen.

Die Experimente dieser Arbeit sind in einzelne Versuchsreihen aufgeteilt. Eine Übersicht

Tabelle 4.1: Übersicht über die durchgeführten Experimente und deren Parameterkombinationen. Jede Versuchsreihe wurde mit 20 verschiedenen Seeds durchgeführt. Wird ein Versuch nochmals mit dem gleichen Seed gestartet, so werden die Ergebnisse gleich sein, weil über Kongruenzgeneratoren in Java lediglich Pseudo-Zufallszahlen erzeugt werden.

Versuchsreihe	wolfSpeed	numberOfDistanceSensors	brainMutationInterval
1/18	1.0	3	50
2/18	1.2	3	50
3/18	1.5	3	50
4/18	1.0	6	50
5/18	1.2	6	50
6/18	1.5	6	50
7/18	1.0	3	100
8/18	1.2	3	100
9/18	1.5	3	100
10/18	1.0	6	100
11/18	1.2	6	100
12/18	1.5	6	100
13/18	1.0	3	200
14/18	1.2	3	200
15/18	1.5	3	200
16/18	1.0	6	200
17/18	1.2	6	200
18/18	1.5	6	200

über alle durchgeführten Experimente befindet sich in Tabelle 4.1.

## 4.2 Auswertung/Ergebnisse

Schaubilder wurden zu allen Versuchsläufen angefertigt und befinden sich auf beiliegender DVD. Diesen Kurven zufolge hat sich in keinem Versuch eine Strategie entwickelt, die langfristig gesehen zu einer höheren Überlebenswahrscheinlichkeit der Beutetiere führt. Weder ist ein dauerhafter Anstieg des Kennwerts „Durchschnittliche Anzahl Schafe in definiertem Radius um ein Schaf“ auf ein höheres Niveau zu verzeichnen, noch lässt sich ein Absinken des Werts „Anzahl gefangener Schafe“ auf ein niedrigeres Level beobachten. In wenigen Versuchen haben sich jedoch zumindest temporäre Beeinflussungen der Gruppenbildungsindikatoren bemerkbar gemacht, die Hinweise auf ein evolviertes Verhalten im Sinne der Aufgabenstellung geben können. Zwei dieser Versuche werden nun genauer vorgestellt. In Versuch 279 aus Versuchsreihe 14 (Abbildung 4.1) kann im Intervall [55.000;75.000] ein Anstieg der Kurve des gleitenden Durchschnitts der durchschnittlichen Anzahl Schafe in definiertem Radius um ein Schaf verzeichnet werden, der einhergeht mit einem Absinken des gleitenden Durchschnitts der Anzahl gefangener Schafe. Dies führt sogar soweit, dass die beiden Kurven ihr globales Maximum bzw. Minimum in diesem Intervall erreichen. Auffällig sind in diesem Intervall neben den gleitenden Durchschnitten auch der einfache Durchschnitt, der leicht erhöht ist und das Maximum, das zwar nicht erhöht ist, aber weniger stark nach unten schwankt. Eine Betrachtung des entsprechenden Trajektorienbildes der Zyklen [60.010;70.000] veranschaulicht die entstandene Situation: Räuber befinden sich größtenteils in der oberen, Beute jedoch in der unteren Hälfte des Spielfelds (Abbildung 4.2).

In Versuch 202 aus Versuchsreihe 11 (Abbildungen 4.3 und 4.4)) lässt sich im Zyklenintervall [200.000;240.000] eine ähnliche Beobachtung machen. Im Schaubild „Durchschnittliche Anzahl Schafe in bestimmtem Radius um ein Schaf“ steigt der Durchschnitt für ca. 10.000 Zyklen auf ein leicht erhöhtes Level. Diese Entwicklung geht einerseits auch mit einer Erhöhung des Maximums in diesem Intervall einher und schlägt sich andererseits mit

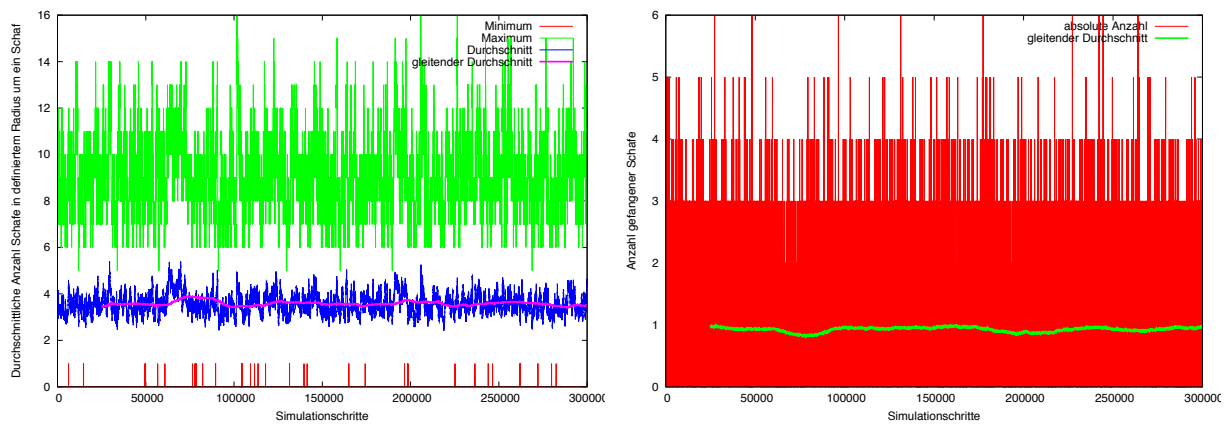


Abbildung 4.1: Schaubilder aus Versuch 279, Versuchsreihe 14.

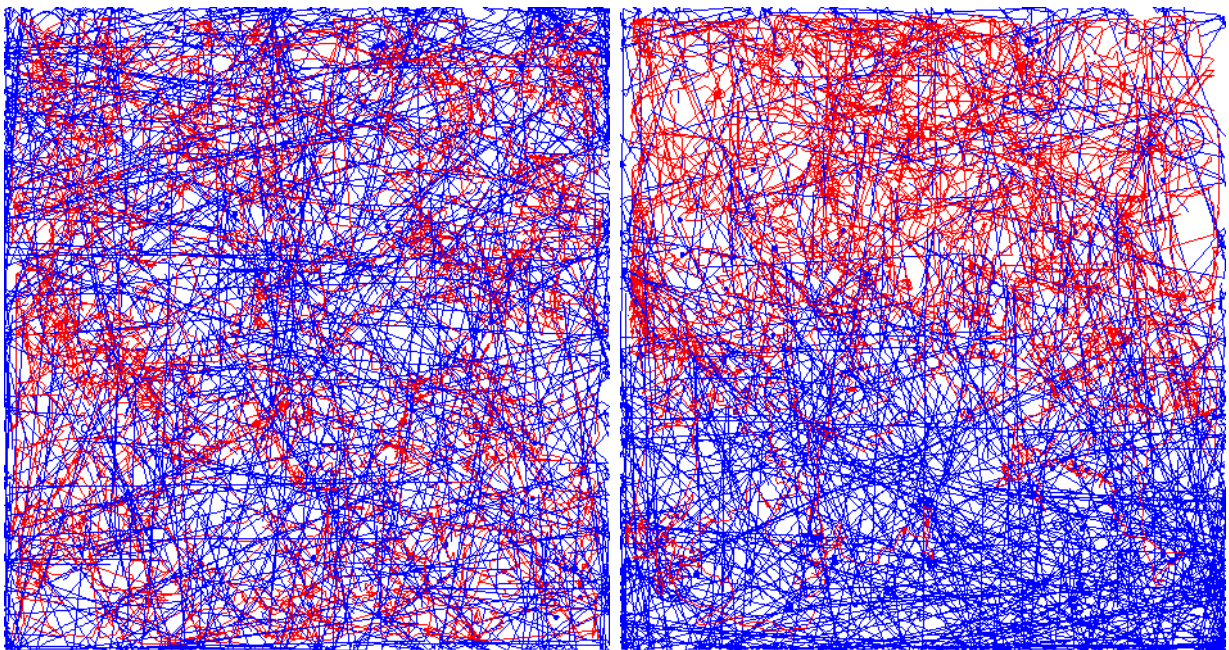


Abbildung 4.2: Trajektorienbilder aus Versuch 279, Versuchsreihe 14. Links das Zyklusintervall  $[50.010; 60.000]$ , rechts das Zyklusintervall  $[60.010; 70.000]$ . Im rechten Bild ist zu erkennen, dass sich Räuber vor allem in der oberen, Beute insbesondere in der unteren Hälfte des Spielfeldes aufhalten. Diese Entwicklung lässt sich nochmals anhand der Videos auf der beiliegenden DVD nachvollziehen.

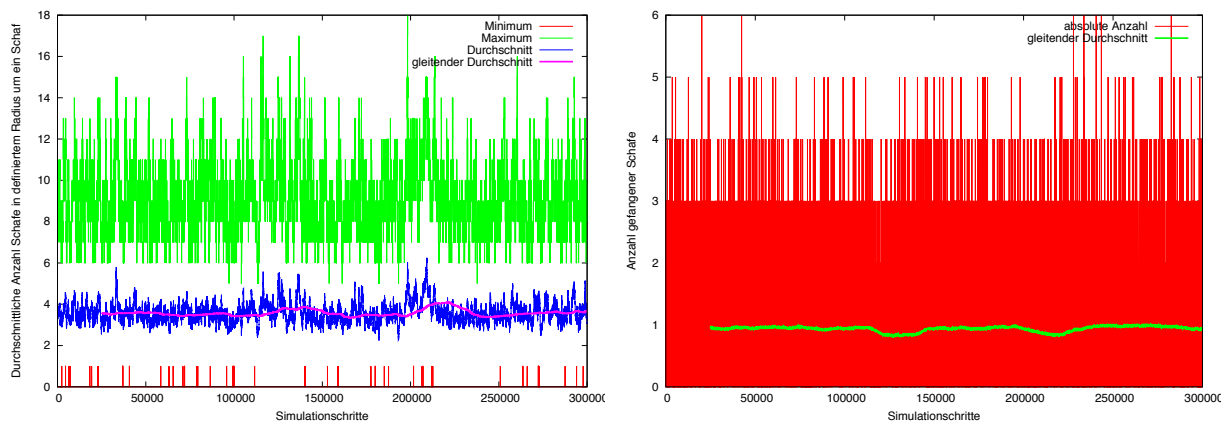


Abbildung 4.3: Schaubilder aus Versuch 202, Versuchsreihe 11.

wenigen tausend Zyklen auch auf den gleitenden Durchschnitt durch, der dadurch sein globales Maximum innerhalb dieses Versuchslaufs annimmt. Gleichzeitig verhält sich der gleitende Durchschnitt dieses Werts im betrachteten Intervall in etwa reziprok zur Anzahl gefangener Schafe, der in diesem Intervall ein lokales Minimum annimmt, das nahe dem globalen Minimum der Kurve ist. Eine Erhöhung der Anzahl Schafe in einem bestimmten Umkreis um ein Schaf geht somit einher mit einer sinkenden Wahrscheinlichkeit des Gefangenwerdens dieses Schafs. Dies scheint die Annahme zu bestätigen, dass eine hohe Anzahl von Schafen um einen Wolf tatsächlich dazu führt, dass der Wolf insgesamt weniger Beutetiere erlegt, weil er ständig das Ziel wechselt. Am Ende dieses betrachteten Intervalls sinken bzw. steigen die entsprechenden Werte wieder auf ihr ursprüngliches Niveau.

Bei den oben beschriebenen und weiteren Versuchen, in denen eine vergleichbare Entwicklung beobachtet werden kann, handelt es sich um Läufe mit einem Mutationsintervall von 100 oder 200. Ein Problem könnte darin bestehen, dass Netze, die eine an die Aufgabenstellung angepasste Verhaltensweise evolviert haben, nicht vor weiterer Mutation geschützt werden. Dadurch könnten sie die erlernte Verhaltensweise wieder verlieren. Daher treten (temporäre) Auffälligkeiten wie in den beiden obigen Versuchen beschrieben

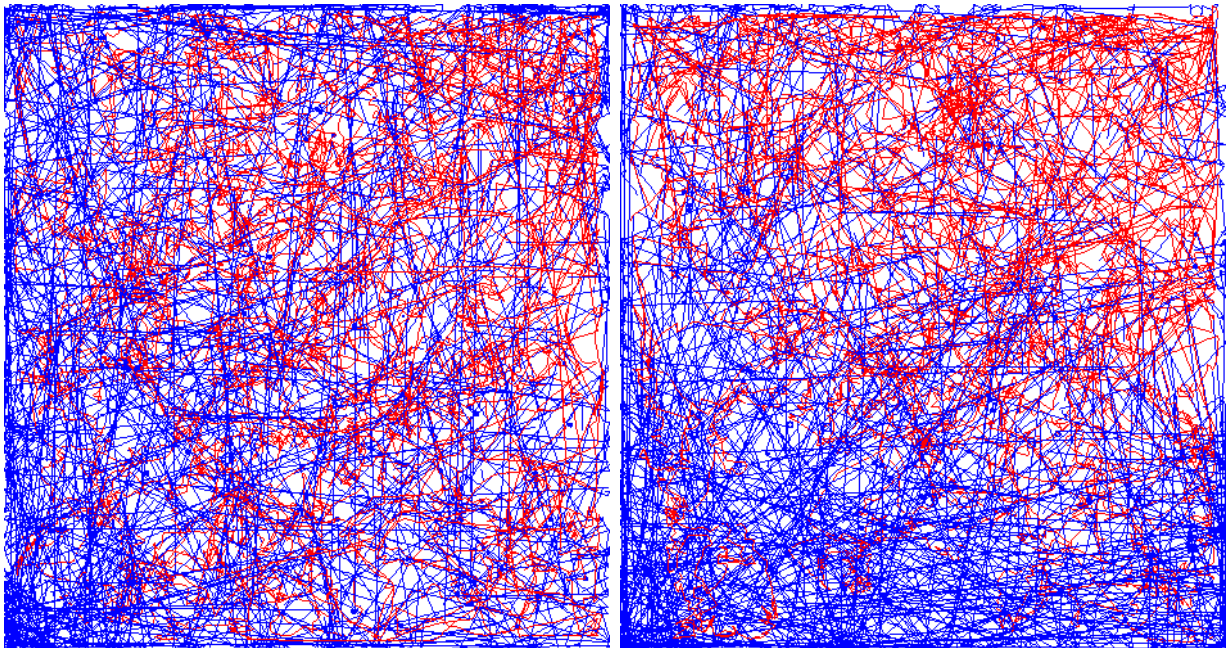


Abbildung 4.4: Trajektorienbilder aus Versuch 202, Versuchsreihe 11. Links das Zyklusintervall  $[190.010; 200.000]$ , rechts das Zyklusintervall  $[200.010; 210.000]$ . Im rechten Bild ist zu erkennen, dass sich Räuber vor allem in der rechten oberen Ecke, Beutetiere insbesondere in der unteren linken Ecke des Spielfeldes aufhalten. Im Vergleich zu Versuch 279 scheint die Veränderung vom linken zum rechten Bild weniger deutlich zu sein. Dies kann jedoch darin begründet liegen, dass die „Spielfeldaufteilung“ zwischen Räuber und Beute bereits im ersten Zyklusintervall beginnt.

nur bei höheren Mutationsintervallen auf. Eine mögliche Lösung dieses Problems könnte in der Nutzung des Parameters „minNeuralNetLifetime“ liegen. Dieser Parameter gibt die minimale Lebensdauer eines neuronalen Netzes an, nach deren Ablauf das neuronale Netz potentiell an andere Roboter weitergegeben werden kann. Dieser Parameter war in den durchgeführten Versuchen auf den Wert 0 eingestellt.

Der Parameter für die Geschwindigkeit des Wolfs („wolfSpeed“) hat in den Experimenten einen systematischen Einfluss auf die Anzahl der gefangenen Schafe: Aus einer höheren Geschwindigkeit der Räuber relativ zur Beute folgt, dass mehr Beute in gleicher Zeit

erreichbar ist und gefangen werden kann. Dies geht sogar soweit, dass dieser Parameter aus dem Diagramm „Anzahl gefangener Schafe“ relativ zuverlässig geschätzt werden kann. Weiter treten die oben beschriebenen Phänomene der Trennung der Individuen in Räuber- und Beutegruppe auf dem Spielfeld insbesondere dann auf, wenn der Wolf nur geringfügig schneller (Faktor 1.2) ist als das Schaf. Eine höhere Geschwindigkeit wird es schwieriger machen, eine adäquate Strategie gegen die Räuber zu entwickeln. Die Distanzsensoren hatten keine erkennbaren Einfluss auf die Ausgänge der Experimente.

Eventuell lernen (größere) Teilmengen der Gesamtpopulation ein erwünschtes Verhalten, welches möglicherweise nicht durch die rein gesamtpopulationsbasierte Auswertungsstrategie erkannt werden kann. Die positiven Auswirkungen einer erlernten positiven Strategie einer Teilpopulation könnten durch das Verhalten der verbleibenden Teilgruppe relativiert werden und somit im Schaubild nicht mehr erkennbar sein. Um solche Fälle zu erkennen, bietet sich eine detaillierte, jedoch zeitintensive Untersuchung von Kombinationen aus Schaf-Trajektorien an.

Möglichweise lassen sich die neuronalen Netze zu den interessanten Zeitpunkten auch analysieren. In den Experimenten stehen aus Speicher- und Zeitgründen nur die Netze am Ende jedes Simulationslaufs zur weiteren Analyse zur Verfügung. Diese bestehen auch aufgrund des verwendeten Mutationsoperators, bei dem Einfügeoperationen eine geringfügig niedrigere Wahrscheinlichkeit haben als Löschooperationen, aus größtenteils wenigen inneren Neuronen, sodass im Einzelfall eine Analyse möglich scheint (vgl. Abbildung 4.5).

Um die Ergebnisse weiter abzusichern und auszuschließen, dass die Beobachtungen rein zufällig sind, sollten Signifikanztests durchgeführt werden. Erst dann wird eine detailliertere abschließende Bewertung des hier identifizierten Verhaltens möglich sein.

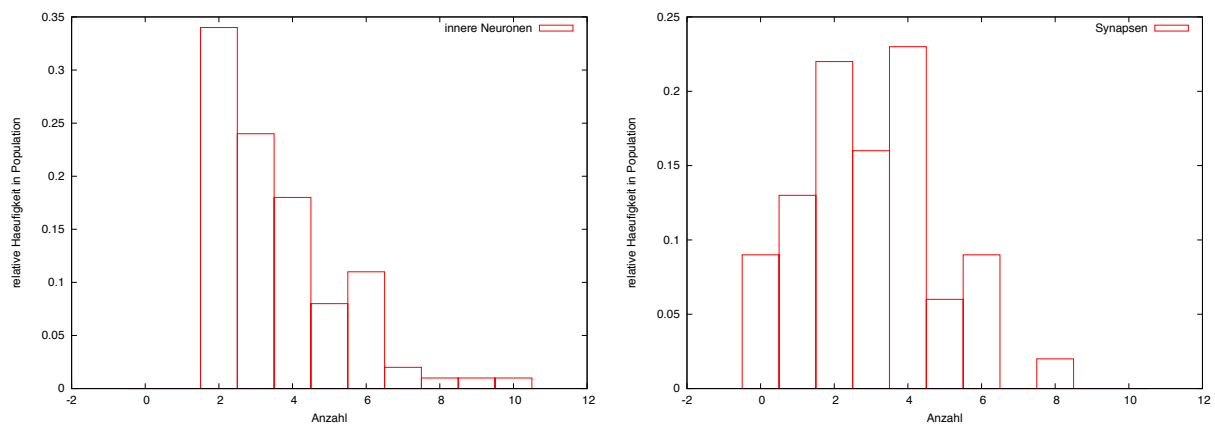


Abbildung 4.5: Relative Häufigkeitsverteilung der Anzahl von Synapsen und von inneren Neuronen. Die Positionen der Synapsen im neuronalen Netzwerk wurden nicht unterschieden. Da sich die Anzahl von inneren Neuronen und von Synapsen noch in Grenzen hält, ist eventuell eine Analyse der Netze möglich (insbesondere dann, wenn keine rekurrenten Verbindungen auftreten).

# Kapitel 5

## Zusammenfassung und Ausblick

### 5.1 Zusammenfassung

Die vorangegangene Arbeit hat sich mit der Evolution von Kooperation im Kontext einer Räuber-Beute-Beziehung beschäftigt. Kooperation wurde in diesem Zusammenhang als Bildung eines Schwarms, also einer Gruppe bestehend aus mehreren Beuterobotern, verstanden. Dabei lag der Fokus auf der Identifizierung von Faktoren, welche sich positiv auf eine solche Schwarmbildung auswirken. Als Umgebung wurde ein ca. 500 x 500 WLE<sup>2</sup> großes Feld ohne Hindernisse (bis auf die vier Wände) gewählt, in dem sich Räuber und Beute befanden. In den Simulationen hat sich keine Strategie entwickelt, die zu einer dauerhaften Veränderung der Überlebenswahrscheinlichkeit der Beute führt. Es haben sich jedoch Hinweise auf eine temporäre Strategiefindung ergeben, die noch einer Signifikanzprüfung unterzogen werden müssen, um abschließend beurteilt werden zu können. Erwartungsgemäß hat sich der implizite Fitnessansatz als Schwierigkeit herausgestellt. Dieser wurde allerdings bewusst gewählt, weil die Natur eine entsprechende Fitnessfunktion nicht kennt und dieser implizite Fitnessansatz der Natur am nächsten kommt. Diese löste und löst fortwährend im Laufe der Evolution hochgradig mehrdimensionale nichtli-

neares Optimierungsprobleme — und dies ganz ohne Fitnessfunktion!

Insgesamt haben sich die Versuche als zeitaufwändig erwiesen, sodass eindeutige Ergebnisse wohl nur mit großem Aufwand zu entdecken sind. Um die Wirkung der Parameter genau zu verstehen, wäre es wünschenswert, einzelne Parameter in der Evolution noch während des Simulationslaufs verändern zu können. Andernfalls gestaltet sich die Suche nach vielversprechenden Startparametern (das gilt natürlich nur dann, wenn noch keine Erfahrungswerte in der verwendeten Umgebung existieren) sehr zeitintensiv.

Im Übrigen wurde in dieser Arbeit Wert auf die Wiederverwendung des Räuber-Beute-Szenarios im EAS-Framework gelegt, so dass sich zukünftig noch weitere Studien damit durchführen lassen. Dabei kann auch ein im Zuge dieser Arbeit mitimplementiertes Statistikmodul zur Aufzeichnung und Abspeicherung von diversen Simulationsdaten nebst grafischer Visualisierung der Robotertrajektorien hilfreich sein. Durch Serialisierung von einzelnen neuronalen Netzen wurde ferner eine Möglichkeit geschaffen, evolvierte neuronale Netze persistent zu machen, zu speichern und in weiteren Tests bzw. anderen Szenarien zu verwenden.

## 5.2 Ausblick

Die vorliegende Arbeit kann aufgrund des vorgegebenen zeitlichen und inhaltlichen Umfangs nur einzelne Aspekte des Themengebiets abdecken. So kann die vorliegende Implementierung nach Erweiterung der Strategie des Räubertieres (Wolf) für die Untersuchung von Koevolution mit neuronalen Netzen als Steuerprogramm für die Individuen herangezogen werden. Gerade in diesem Zusammenhang ist die Einführung eines „Verhungereintervalls“ der Räuber denkbar. Durch die Miteinbeziehung einer Fitness-Funktion könnte man versuchen, die Evolution zu kanalisieren (also in eine bestimmte Richtung zu lenken), womit Ergebnisse unter Umständen schon nach weniger Simulationszyklen feststellbar sein könnten. Damit verlässt man allerdings den eingeschlagenen Pfad dieser

Untersuchung, deren besonderer Reiz gerade die Verwendung von impliziter Fitness war. Auch ist die Untersuchung in einer komplexeren Umgebung mit einzelnen Hindernissen möglich.

Weitere Betätigungsmöglichkeiten ergeben sich, wenn man zwischen Kooperation – d.h. schwachem Altruismus (da das kooperierende Individuum keinen Fitnessverlust erleidet) – auf der einen und der starken Form des Altruismus (bei der das sich altruistisch verhaltende Individuum einen echten Fitnessverlust erleidet) auf der anderen Seite differenziert. Als Vorbild könnten hier wiederum Verhaltensweisen von Tieren in der Natur herangezogen werden (z.B. das Warnen der eigenen Gruppe von Individuen vor einem Räuber auf Kosten der Preisgabe der eigenen Position, womit eine höhere Wahrscheinlichkeit des Gefangenwerdens einhergeht). Dazu könnte auch die Implementierung von Kommunikationssensoren sinnvoll sein.

In der Natur zeigt sich, dass altruistische Verhaltensweisen vor allem zwischen genetisch ähnliche Individuen anzutreffen sind. Dies könnte als Inspiration genommen werden, einen Vergleich anzustellen zwischen der Bereitschaft für altruistisches Handeln innerhalb von Gruppen, die ihren eigenen Gruppenmitgliedern ähnlicher sind als den Gruppenmitgliedern der jeweils anderen Gruppe [WKF09].

Interessant wäre auch die Implementierung weiterer Mutations- und Rekombinationsoperatoren. So könnte man an der Darstellung des Genotyps eines Roboters bzw. neuronalen Netzes als Genom arbeiten, die sinnvolle Rekombinationsoperationen auf Genom-Ebene möglich macht.



# Anhang A

## Parametersätze

Tabelle A.1: Allgemeiner Standardparametersatz. Parameter mit \* variieren in den einzelnen Experimenten.

Parameter	Wert	Erläuterung
explen	300000	Experimentlänge in Zyklen. (auch: ticks)
verzögerung	0	Verzögerung zwischen den Simulationszyklen.
masterScheduler	defaultmaster-co	Kennung des verwendeten Master-Schedulers.
plugin	zeitschaetzer, trajectoryStatistics	Verwendete Erweiterungen. <i>zeitschaetzer</i> schätzt die verbleibende Zeit bis zum Abschluss des Simulationslaufs. <i>trajectoryStatistics</i> zeichnet statistische Daten auf.
seed	203126278773960577*	Seed für Zufallsgenerator.

Tabelle A.2: Spezieller Parametersatz des Master-Schedulers. Parameter mit \* variieren in den einzelnen Experimenten.

Parameter	Wert	Erläuterung
numberOfSheep	100	Anzahl der Schafe (Beuteroboter).
numberOfWolf	20	Anzahl der Wölfe (Räuberroboter).
wolfDetectionRadius	50.0	Beute-Erkennungsradius der Räuber.
wolfCatchRadius	5.0	Beute-Fangradius der Räuber.
sheepSpeed	1.0	Maximalgeschwindigkeit der Schafe (Beuteroboter).
wolfSpeed	1.5*	Maximalgeschwindigkeit der Wölfe (Räuberroboter).
numberOfDistanceSensors	3*	Anzahl der Distanzsensoren der Schafe (Beuteroboter).
chooseSheepInterval	20	Zeitfenster (in ticks), nachdem der Wolf (Räuberroboter) ein neues Schaf (Beuteroboter) auswählt.

Tabelle A.3: Spezieller Parametersatz des neuronalen Netzes (Klasse NeuroBrain). Parameter mit \* variieren in den einzelnen Experimenten.

Parameter	Wert	Erläuterung
brainSize	3	Anfängliche Anzahl der versteckten Neuronen des künstlichen neuronalen Netzes. Verändert sich dynamisch im Laufe der Evolution.
brainMutationInterval	50*	Zeitfenster, nach dessen Ablauf das neuronale Netz mutiert (in ticks).
minNeuralNetLifetime	0	Minimale Lebensdauer des neuronalen Netzes, nach deren Ablauf das neuronale Netz potentiell an andere Roboter weitergegeben werden kann.

Tabelle A.4: Spezieller Parametersatz des Statistik-Moduls. Parameter mit \* variieren in den einzelnen Experimenten

Parameter	Wert	Erläuterung
startTrace	0	Beginn der Aufzeichnungen (in ticks).
stopTrace	300000	Ende der Aufzeichnungen (in ticks).
agentsToTrace	all	Zu welchen Agenten statistische Daten aufgezeichnet werden sollen. Mögliche Werte: <i>all</i> , <i>sheep</i> und <i>wolf</i>
recordingInterval	10	Aufzeichnungsintervall. Gespeichert wird unabhängig davon in jedem Zyklus.
saveTrajectoriesEach	10000	Nach wie vielen Zyklen ein Bild der Trajektorien im png-Format gespeichert werden soll. Trajektorien der Schafe werden blau, diejenigen der Wölfe rot dargestellt.
fileDestination	output000*	Beginn des Dateinamens/Ordners, in dem gespeichert wird (ausgehend vom angegebenen Arbeitsverzeichnis).
calculateStatisticsEachKTicks	100	In welchen Intervallen jeweils statistische Kennwerte berechnet werden.



# Literaturverzeichnis

- [AB97] AGAH, Arvin ; BEKEY, George A.: Phylogenetic and Ontogenetic Learning in a Colony of Interacting Robots. In: *Autonomous Robots* 4 (1997), Nr. 1
- [AH81] AXELROD, Robert ; HAMILTON, William D.: The Evolution of Cooperation. In: *Science. New Series* 211 (1981), Nr. 4489, S. 1390–1396
- [CR06] CAMPBELL, Neil A. ; REECE, Jane B.: *Biologie*. 6. Auflage. München [u.a.] : Pearson Studium, 2006 (Bio - Biologie)
- [Dar59] DARWIN, Charles: *On the Origin of Species – A Facsimile of the First Edition*. Cambridge : Harvard University Press, 1859
- [DK79] DAWKINS, Richard ; KREBS, John R.: Arms Races between and within Species. In: *Proceedings of the Royal Society of London. Series B, Biological Sciences* 205 (1979), Nr. 1161, S. 489–511
- [Fah90] FAHLMAN, Scott E.: The recurrent cascade-correlation architecture. In: *NIPS-3: Proceedings of the 1990 conference on Advances in neural information processing systems 3*, Morgan Kaufmann Publishers Inc., 1990, S. 190–196
- [FL90] FAHLMAN, Scott E. ; LEBIERE, Christian: The Cascade-Correlation Learning Architecture / School of Computer Science, Carnegie Mellon University. Pittsburgh, PA : Morgan Kaufmann, 1990 (CMU-CS-90-100). – Forschungsbericht

- [Ham64] HAMILTON, William D.: The Genetical Evolution of Social Behavior. In: *Journal of Theoretical Biology* 7 (1964), Nr. 1, S. 1–52
- [HTP<sup>+</sup>06] HOEN, Pieter ; TUYLS, Karl ; PANAIT, Liviu ; LUKE, Sean ; LA POUTRÉ, J.A.: An Overview of Cooperative and Competitive Multiagent Learning. In: *Learning and Adaption in Multi-Agent Systems* Bd. 3898. 2006, S. 1–46
- [Hyo96] HYOTYNIEMI, Heikki: Turing machines are recurrent neural networks. (1996)
- [Jan80] JANZEN, Daniel H.: When is it Coevolution? In: *Evolution* 34 (1980), Nr. 3, S. 611–612
- [KEH00] KOECKE, Hans-Ulrich ; EMSCHERMANN, Peter ; HÄRLE, Eckhart: *Biologie : Lehrbuch der allgemeinen Biologie für Mediziner und Naturwissenschaftler; 22 Tabellen*. 4. Auflage. Stuttgart [u.a.] : Schattauer, 2000
- [KMS09] KÖNIG, Lukas ; MOSTAGHIM, Sanaz ; SCHMECK, Hartmut: Online and On-board Evolution of Robotic Behavior Using Finite State Machines (Extended Abstract). In: *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009)* (2009), S. 1325–1326
- [KS08] KÖNIG, Lukas ; SCHMECK, Hartmut: Evolving Collision Avoidance on Autonomous Robots. In: *Biologically-Inspired Collaborative Computing* Bd. 268. 2008, S. 85–94
- [Lam09] LAMARCK, Jean-Baptiste de: *Philosophie zoologique, ou, Exposition des considerations relative a l'histoire naturelle des animaux*. Paris : Chez Dentu [et] L'Auteur, 1809
- [Men66] MENDEL, Gregor: Versuche über Pflanzenhybriden. In: *Verhandlungen des Naturforschenden Vereines in Brünn* IV (1866), S. 3–47

- [MP43] MCCULLOCH, Warren ; PITTS, Walter: A logical calculus of the ideas immanent in nervous activity. In: *Bulletin of Mathematical Biology* 5 (1943), Nr. 4, S. 115–133
- [Mü10] MÜLLER, Benedikt: *Neuroevolution in Roboterschwärmen. Einführungsvortrag zu neuronalen Netzen*. 2010
- [NF01] NOLFI, Stefano ; FLOREANO, Dario: *Evolutionary robotics : the biology, intelligence, and technology of self-organizing machines*. 2. print. Cambridge, Massachusetts : MIT Press, 2001 (Intelligent robots and autonomous agents)
- [PL05] PANAIT, Liviu ; LUKE, Sean: Cooperative Multi-Agent Learning: The State of the Art. In: *Autonomous Agents and Multi-Agent Systems* 11 (2005), Nr. 3, S. 387–434
- [PMS01] POTTER, Mitchell A. ; MEEDEN, Lisa ; SCHULTZ, Alan C.: Heterogeneity in the Coevolved Behaviors of Mobile Robots: The Emergence of Specialists. In: *Proceedings of the 17th international joint conference on Artificial intelligence*, 2001, S. 1337–1343
- [SKM10] SCHMECK, Hartmut ; KÖNIG, Lukas ; MOSTAGHIM, Sanaz: *Skript zu Organic Computing: Learning Robots*. 2010
- [SM02] STANLEY, Kenneth O. ; MIIKKULAINEN, Risto: Evolving neural networks through augmenting topologies. In: *Evolutionary Computation* 10 (2002), S. 99–127
- [Tri71] TRIVERS, Robert L.: The Evolution of Reciprocal Altruism. In: *The Quarterly Review of Biology* 46 (1971), Nr. 1, S. 35–57

- [WD08] WÖHE, Günter (Hrsg.) ; DÖRING, Ulrich (Hrsg.): *Vahlens Handbücher der Wirtschafts- und Sozialwissenschaften*. Bd. [Hauptbd.]:: *Einführung in die allgemeine Betriebswirtschaftslehre*. 23. Auflage. München : Vahlen, 2008
- [Wei99] WEICKER, Karsten: Evolutionäre Algorithmen. In: *Softcomputing - Tagungsband zum ersten Softcomputing-Treffen*, 1999, S. 27–39
- [Wei07] WEICKER, Karsten: *Evolutionäre Algorithmen*. 2. Auflage. Leipzig : B.G. Teubner Verlag / GWV Fachverlage GmbH, 2007 (Leitfäden der Informatik)
- [WGW03] WALKER, Joanne ; GARRETT, Simon ; WILSON, Myra: Evolving controllers for real robots: A survey of the literature. In: *Adaptive Behavior* 11 (2003), Nr. 3, S. 179–203
- [WKF09] WAIBEL, Markus ; KELLER, Laurent ; FLOREANO, Dario: Genetic Team Composition and Level of Selection in the Evolution of Cooperation. In: *IEEE Transactions on Evolutionary Computation* 13 (2009), Nr. 3, S. 648–660
- [ZC99] ZHANG, Byoung-Tak ; CHO, Dong-Yeon: Co-evolutionary fitness switching: learning complex collective behaviors using genetic programming. In: *Advances in genetic programming* 3 (1999), S. 425–445

## **Erklärung**

Ich versichere wahrheitsgemäß, die Arbeit selbstständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderungen entnommen wurde.

---

Karlsruhe, den 14. September 2010, Christian Nagel

## DVD

Auf dieser DVD befinden sich

- das den Versuchen zugrunde liegende EAS-Framework,
- die JoSchKa-Pakete (diese werden zur Verteilung der Versuchsläufe auf die einzelnen Rechner benötigt),
- die aufgezeichneten Versuchsdateien (CSV-Format), Trajektorienbilder (PNG-Format), Videos (AVI-Format) und die Schaubilder aller 360 Versuche (EPS-Format),
- diese Ausarbeitung nebst aller verwendeten Quellen (soweit als PDF-Datei verfügbar) sowie der Abschlussvortrag.